НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

«До захисту допущено»
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою
«Програмне забезпечення інформаційно-комунікаційних систем»
спеціальності 121 «Інженерія програмного забезпечення»
на тему: «Програма для розпізнавання продуктів та пошуку рецептів з ними на основі комп'ютерного зору та Android SDK»

Виконав:
студент IV курсу, групи ІТ-61
Сердюк Богдан Сергійович _____

Керівник:
доцент каф. АУТС, к.т.н.,
Жереб Костянтин Анатолійович _____

Рецензент:
доцент каф. ТК, к.т.н.,
Ткач Михайло Мартинович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ — 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

Рівень вищої освіти — перший (бакалаврський)

Спеціальність — 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Програмне забезпечення інформаційно-комунікаційних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

ЗАВДАННЯ

на дипломний проєкт студенту

Сердюку Богдану Сергійовичу

1. Тема проєкту «Програма для розпізнавання продуктів та пошуку рецептів з ними на основі комп'ютерного зору та Android SDK», керівник проєкту Жереб Костянтин Анатолійович, доцент кафедри АУТС, затверджені наказом по університету від «7»травня 2020 р. № 1081-с

2. Термін подання студентом проєкту: 9.06.2020

3. Вихідні дані до проєкту: мова програмування Kotlin, середовище розробки Android Studio.

4. Зміст пояснювальної записки: аналіз предметної області, огляд існуючих рішень, проєктування та реалізація власного рішення.

5. Перелік графічного матеріалу: діаграма класів, діаграма пакетів, діаграма активності, діаграма прецедентів.

6. Дата видачі завдання 3 березня 2020 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення рекомендованої літератури	10.04.2020	Виконано
2.	Аналіз теоретичних матеріалів та вивчення предметної області	13.04.2020	Виконано
3.	Аналіз існуючих рішень	20.04.2020	Виконано
4.	Формування вимог до додатку	24.04.2020	Виконано
5.	Розробка структури програмного забезпечення	26.04.2020	Виконано
6.	Написання програмних модулів додатку та інтерфейсів взаємодії з віддаленим та локальним API	29.04.2020	Виконано
7.	Розроблення структури та проєктування бази даних збережених рецептів користувача	14.05.2020	Виконано
8.	Підключення віддалених API та локальної бази даних рецептів користувача через інтерфейси взаємодії	17.05.2020	Виконано
9.	Оформлення пояснювальної записки	18.05.2020	Виконано
10.	Подання ДП на попередній захист	25.05.2020	Виконано
11.	Подання ДП рецензенту	08.06.2020	Виконано
12.	Подання ДП на основний захист	09.06.2020	Виконано

Студент Богдан СЕРДЮК

Керівник Костянтин ЖЕРЕБ

АНОТАЦІЯ

Сердюк Б.С. Програма для розпізнавання продуктів та пошуку рецептів з ними на основі комп’ютерного зору та Android SDK. КПІ ім. Ігоря Сікорського, Київ, 2020.

Пояснювальна записка викладена на 65 сторінці, містить 3 розділи, 32 рисунків, 7 таблиць та 35 джерел.

Ключові слова: комп’ютерний зір, Android додаток.

Об’єктом розробки є програма для розпізнавання продуктів та пошуку рецептів з ними на основі комп’ютерного зору та Android SDK.

Мета розробки – знайти оптимальне рішення для покращення способу введення запиту користувачем.

У дипломному проєкті досліджено переваги та недоліки існуючих рішень, що здійснюють пошук рецептів за інгредієнтами, розроблене власне рішення, що враховує недоліки існуючих рішень та покращує досвід роботи з додатком, що отримує користувач. Розроблена повноцінна архітектура додатку, що дозволить швидко та зручно реалізовувати новий функціонал та розширювати кількість підтримуваних платформ. Проведено ретельне дослідження та порівняння сервісів необхідних для повноцінної роботи додатку. Розроблений сучасний дизайн, що відповідає принципам сучасного підходу до UI/UX дизайну. Завдяки нативності реалізації та оптимізації внутрішніх процесів, додаток відповідає вимогам по енерговитратності та навантаженню на CPU.

Отримані результати допоможуть спростити процес пошуку рецептів користувачем та оптимізувати час, що витрачається на формування запиту.

SUMMARY

Bohdan Serdiuk. Program for recognizing products and finding recipes with them based on computer vision and Android SDK. Igor Sikorsky Kyiv Polytechnic Institute, Kyiv 2020.

The explanatory note is set out on 65 pages, containing 3 sections, 32 figures, 7 tables and 35 sources.

Keywords: computer vision, Android application.

The object of development is a program for product recognition and recipes searching based on computer vision and Android SDK.

The purpose of development is to find the optimal solution to improve the way of users query entering.

The diploma project explores advantages and disadvantages of existing solutions that search for recipes by ingredients, developed its own solution that takes into account the shortcomings and improves users experience. A full-fledged application architecture has been developed, which will allow quickly and easily implement new functionality and expand the number of supported platforms. A thorough study and comparison of services required for the full operation of the application. Developed design that meets the principles of a modern approach to UI / UX design. Due to the native implementation and optimization of internal processes, the application meets the requirements for power consumption and CPU load.

The results will help simplify the process of finding recipes by the user and optimize the time spent on the formation of the query.

Поз.	Формат	Позначення	Найменування	Кількість аркушів	№ прим.	Примітки
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	А4	ІТ61.190БАК.004 ПЗ	Програма розпізнавання	65		
6			продуктів			
7			та пошуку рецептів на основі			
8			комп’ютерного зору			
9			Пояснювальна записка			
10	А3	ІТ61.190БАК.004 Д1	Програма розпізнавання	1		
11			продуктів			
12			та пошуку рецептів на основі			
13			комп’ютерного зору			
14			Діаграма класів			
15	А3	ІТ61.190БАК.004 Д2	Програма розпізнавання	1		
16			продуктів			
17			та пошуку рецептів на основі			
18			комп’ютерного зору			
19			Діаграма пакетів			
20	А3	ІТ61.190БАК.004 Д3	Програма розпізнавання	1		
21			продуктів			
22			та пошуку рецептів на основі			
23			комп’ютерного зору			
24			Діаграма активності			
25	А3	ІТ61.190БАК.004 Д4	Програма розпізнавання	1		
26			продуктів			
27			та пошуку рецептів на основі			
28			комп’ютерного зору			
29			Use Case діаграма			
30						
31						
32						
33						
34						
35						
36						

					ІТ61.190БАК.004 ТП		
Змн.	Лист	№ докум.	Підпис	Дат			
Розроб.		Сердюк Б.С.			Програма для розпізнавання продуктів та пошуку рецептів з ними на основі комп’ютерного зору та Android SDK. Відомість технічного проєкту	Літ.	Лист 1 / Листів 1
Перевір.		Жереб К.А.					
						КПІ ім. Ігоря Сікорського кафедра АУТС гр. ІТ-61	
Н. Контр.							
Затверд.		Ролік О.І.					

Пояснювальна записка

до дипломного проєкту

на тему: «Програма для розпізнавання продуктів та пошуку рецептів з ними на основі комп’ютерного зору та Android SDK»

Київ – 2020 року

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

ЗНМ – Згорткова нейронна мережа

СУБД – Система управління базами даних

ANR – Application Not Responding

API – Application Programming Interface

CA – Clean Architecture

Code sharing – можливість повторного використання коду в інших проєктах

DTO – Data Transfer Object

MD – Material Design

MLaaS – Machine learning as a service

MVP – Model View Presenter

UI/UX – дизайн інтерфейсу та логіка взаємодії користувача з інтерфейсом

ВСТУП

На сьогоднішній день, сфера інформаційних технологій є основним рушієм автоматизації процесів, що не потребують постійної людської уваги. Велика кількість сучасних продуктових ІТ-компаній була створена на основі розробок, які допомагають зменшити або й взагалі замінити людську діяльність на автоматизовані системи.

Однією з невід'ємних сфер людської життєдіяльності є споживання їжі, а отже зацікавленість в шляхах приготування, та отримання якомога доступнішої супровідної інформації залишається актуальною завжди. Більшість програм пов'язаних з цією сферою, що знаходяться на ринку мобільних додатків, допомагають здійснювати пошук за рецептів за інгредієнтами, знаходити та отримувати всю необхідну супровідну інформацію, включаючи інструкції приготування та перелік інгредієнтів.

Тим не менше, додатки, що пропонують рецепти з приготування їжі далекі від повної або часткової автоматизації пошуку та пропонування рецептів за інгредієнтами, в більшості існуючих рішень відсутня підтримка стабільної роботи додатку в офлайні, що негативно впливає на досвід, що отримує користувач при роботі з додатком. Кінцевий користувач зацікавлений у вирішенні проблеми, для якої створений додаток якомога простішим та інтуїтивним чином, при цьому з мінімальною витратою часу. Проблема оптимізації витраченого користувачем часу на формування запиту є нетривіальною, адже всі можливі оптимізації з користувацьким інтерфейсом та стандартними інструментами введення інформації вже були проведенні розробниками до цього.

Для вирішення даної проблеми необхідно прийняти нетривіальний підхід, що допоможе отримати конкурентну перевагу в порівнянні зі схожими

додатками на мобільному ринку. Рішення має бути адаптивним та підтримувати можливість подальшого розширення на інші операційні системи і платформи.

Отриманий продукт матиме конкурентну перевагу відносно існуючих додатків, які пропонують рецепти за списком інгредієнтів, що допоможе зайняти необхідну нішу на ринку та покращити підхід, взаємодії користувача з додатком. Продумана реалізація та оптимізація внутрішніх процесів додатку має задовольниити вимоги по енерговитратності та навантаженню на CPU.

Вирішенням цієї проблеми є використанням останніх технологій автоматизації класифікації із застосуванням моделей нейронних мереж, що допоможуть відійти від стандартних методів введення переліку інгредієнтів за допомогою екранної клавіатури мобільного додатку, на користь формування запиту на основі розпізнаних, за допомогою моделей комп'ютерного бачення та класифікації, інгредієнтів з фотографії зробленої користувачем, що в рази збільшить швидкість запиту користувача та розширить можливості для подальшого розвитку сфери в напрямку штучного інтелекту.

Актуальність теми проєкту: зростання популярності тем машинного навчання та задання пошукової інформації за допомогою технологій комп’ютерного бачення разом зі стабільною популярністю додатків для приготування їжі, визначають цю тему як актуальну.

Об’єкт дослідження: об’єктом дослідження є ринок мобільних додатків, що пропонують рецепти за запитом користувача.

Предмет розробки: предметом розробки є програма для розпізнавання продуктів та пошуку рецептів з ними на основі комп’ютерного зору та Android.

Мета: метою цього проєкту є створення сучасного мобільного додатку, що буде розроблений на основі сформованого переліку переваг та недоліків існуючих рішень у власній реалізації.

Практичне значення: розроблюваний додаток надає користувачу більш інтерактивний підхід до формування запиту та швидкості введення інформації за якою здійснюється пошук.

Положення: Завдяки технології комп'ютерного бачення буде отримана вагома перевага в часі, що витрачається користувачем на введення запиту, в порівнянні з існуючими рішеннями.

Структура проєкту:

— у розділі 1 представлено опис розроблюваного рішення та сформований перелік вимог до проєктованого додатку. Оглянуті найбільш популярні варіанти існуючих рішень;

— у 2 розділі розроблене архітектурне рішення додатку, створені інтрефейси взаємодії з віддаленим API;

— у 3 розділі проведений аналіз та обрані допоміжні сервіси, обрана СУБД

— у 4 розділі обраний дизайн додатку, розроблене рішення протестоване та порівняно з конкурентними розробками.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис проєктованого додатку

Ринок мобільних програм, що пропонують можливості пошуку рецептів за переліком інгредієнтів достатньо різноманітний. Більшість існуючих рішень не оптимізовані за часом, який витрачає користувач на користування ними та отримання інформації, задля якої він цей додаток завантажив, а головне введення самого запиту [1, 2].

Основною перевагою пропонованої реалізації є акцент на безпосередньому використанні моделей машинного навчання, що розпізнають та формують перелік інгредієнтів, на основі фотографії зробленої користувачем, та подальшому пропонуванні рецептів на основі розпізнаних продуктів. Витрата часу при такому підході є неспіврозмірно меншою, порівняно з існуючими розробками, адже лише для написання слова “apple” людині в середньому необхідно приблизно 2 секунди, та список інгредієнтів може продовжуватися, а введення кожного окремо є часовитратною та неоптимальною задачею. За допомогою такого підходу розвантажується інтерфейс додатку, по причині відсутності компонентів, які більше не є затребуваними. Користувач отримує новий інтерактивний досвід, при взаємодії з графічним інтерфейсом додатку.

1.2 Вимоги до проєктованого додатку

Базовими вимогами для реалізованого рішення є максимальна кількість об’єктів, що піддаються класифікації з фотографії, яку робить користувач на власну камеру, перехоплення та сповіщення про помилки під час процесу класифікації запиту користувача, фільтрація інгредієнтів серед розпізнаних при процесі класифікації об’єктів. Сервіс надання послуг по отриманню списку

рецептів має підтримувати можливість формування результатів за обмеженою запитом користувача кількістю інгредієнтів.

Окрім цього, проєктований додаток має базуватися на сучасному архітектурному підході, що забезпечує можливість подальшого оптимального розширення та code sharing між проєктами, підтримувати шляхи міграції на інші операційні системи.

Розроблене рішення має стабільно працювати при відсутності інтернет підключення та загалом. При необхідності, шляхом мануального тестування необхідно визначити найбільш проблемні частини додатку, покрити їх unit тестами упевнитися в їхній стабільності на всіх підтримуваних версіях операційної системи Android. При виникненні нестабільних ситуацій та помилок в роботі додатку, користувач має отримати повну технічну допомогу.

Дизайн додатку має базуватися на логічності та стандартах, що базуються на нативних стилях Android додатків, містити лаконічні компоненти та кольори, що гармонійно виглядатимуть разом, реалізувати базові правила UI/UX дизайну.

1.3 Огляд існуючих рішень

На популярних крамницях застосунків AppStore та GooglePlay існує численна кількість рішень, що частково відповідають поставленим вимогам. Додатки допомагають формувати та здійснювати запити користувача, зберігати рецепти та пропонувати нові за вподобаннями. Деякі з них мають можливість створення власних рецептів і пропонування їх для збереження на сервісі. Таким чином вирішена задача актуалізації та доповнення списку існуючих рецептів самими користувачами. Більшість існуючих популярних програм базуються на однойменних сервісах, які пропонують послуги відкритого API, що формує та фільтрує рецепти, інгредієнти та іншу інформацію за запитом розробника, що

допомагає створювати конкурентне середовище серед розробників сервісів фільтрації рецептів за заданими параметрами.

1.3.1 Tasty

Так в Tasty, достатньо популярному додатку для пошуку рецептів, реалізовано зручний механізм формування запиту користувачем, що базується на основі SearchView компоненту, який поставляється разом з Android SDK. Інтерфейс є простим та загалом логічним для користувача, що вперше відкрив додаток. Сервіс постачання рецептів має велику кількість різноманітних форм та шляхів запиту, що базуються на перерахованих інгредієнтах, калорійності, дієтичності страв.

Серед недоліків користувачі додатку відмічають недоліки роботи відео та аудіо кодеків, проблеми з завантаженням контенту та загальну нестабільність роботи додатку [3] (рис. 1.1).

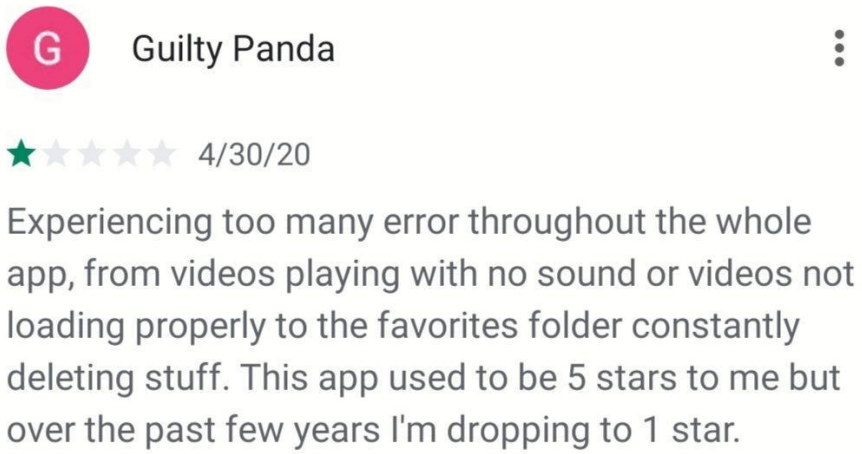

Рисунок 1.1 – Коментар користувача Tasty

Для виведення інформації користувачу, програма використовує компонент WebView, що робить проблематичним: збереження стану додатку при зміні

конфігурацій, змінення теми інтерфейсу або реорганізацію компонентів загалом, без редагування вебпредставлення, імплементацію та кастомізацію відповідно до стандартних вимог дизайну Android додатків на основі Material Design, страждає логічність розміщення компонентів інтерфейсу через те, що додаток працює на основі вебпредставлення, яке не має звичного для користувача Android розміщення компонентів.

Функція збереження рецептів користувачем можлива лише за умови авторизації до Facebook або Google Account без можливості створення акаунту на сервісі, що унеможливлює перегляд та редагування списку при відсутності інтернет з'єднання, створює проблематику користувачам, що не є зареєстрованими або не бажають ділитися конфіденційними даними для розширення функціоналу додатку (рис. 1.2).

Рисунок 1.2 – Коментар користувача Tasty

Швидкість введення даних запиту користувачем цілком залежить від його швидкості набору з екранної клавіатури, тому іноді може бути достатньо часовитратним процесом. Сформуємо список переваг та недоліків на основі аналізу сервісу Tasty.

Переваги:

— механізм формування запиту користувачем;

— простота інтерфейсу;

— сервіс рецептів;

— збереження вподобаних рецептів.

Недоліки:

— кастомізація інтерфейсу;

— логічність розміщення компонентів інтерфейсу;

— обов’язкова авторизація для редагування збережених рецептів;

— відсутність можливості авторизації без прив’язки Facebook або Google Account;

— час введення даних запиту.

1.3.2 GoodFood

Сервіс BBC Good Food є одним із найбільших сховищ різного роду рецептів, надає можливість запитів за великою кількістю параметрів, що є основною перевагою цього рішення. Стандартний для Android застосунків дизайн основних компонентів введення інформації є логічним для кінцевого користувача. Перевагою цього додатку над попереднім рішенням є сторінка рецептів акаунту, що є більш насиченою контентом, на відміну від рішень конкурентів і дозволяє зберігати, а головне створювати нові рецепти, але розділ містить достатньо нелогічну вкладку, що зберігає куховарські книжки, які ніяким чином не пов’язані з вподобаннями кінцевого користувача.

Рішення має власну домашню сторінку, що пропонує користувачу: рецепти за категоріями, останні переглянуті результати запитів, найбільш популярні серед користувачів сервісу рецепти.

Додаток GoodFood має громіздкий контент, що внаслідок його перенасиченості є не зовсім зручним для користувача, який вперше запустив

програму, що позначається на загальному рейтингу додатку в Play Store, він складає всього 3.7 з 5 адже користувачі відмічають його нестабільну роботу [4] (рис. 1.3).

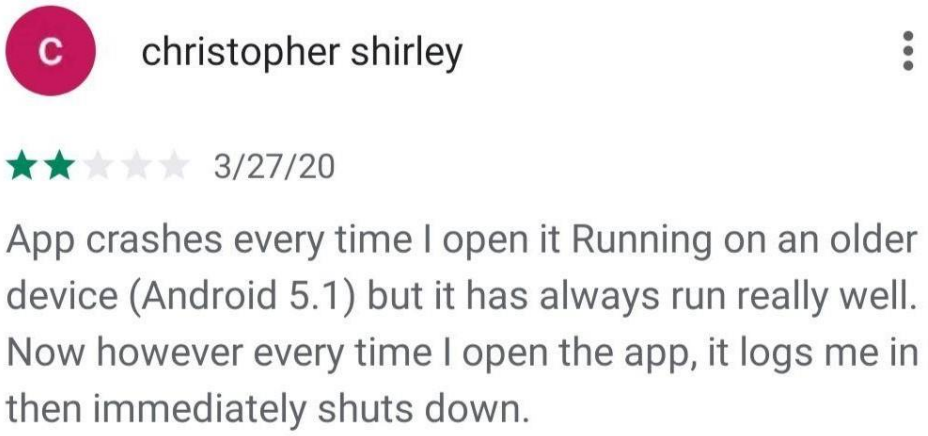

Рисунок 1.3 – Коментар користувача GoodFood

Серед недоліків програми також є відкриття веб-сторінки для виведення всіх можливих результатів запиту користувача, що унеможливлює їх зміну та кастомізацію без редагування вебпредставлення. Так само, як і перше рішення, в GoodFood присутня обмеженість швидкості запиту користувачем, що залежить від вміння набирати текст з клавіатури.

На відміну від попереднього додатку, користувач реєструється безпосередньо на сервісі GoodFood з обов'язковим використання електронної пошти, це унеможливлює конфіденційність та безпечне збереження даних користувача довіреними сервісами та продукує можливість спаму або некоректної роботи програми.

Користувачі додатку інформують про неможливість повторного надсилання листа на електронну пошту при помилковому введені даних або втрати доступу до поштового сервісу, що безумовно є недоліком бізнес-логіки, яка входить до складу рішення (рис. 1.4).

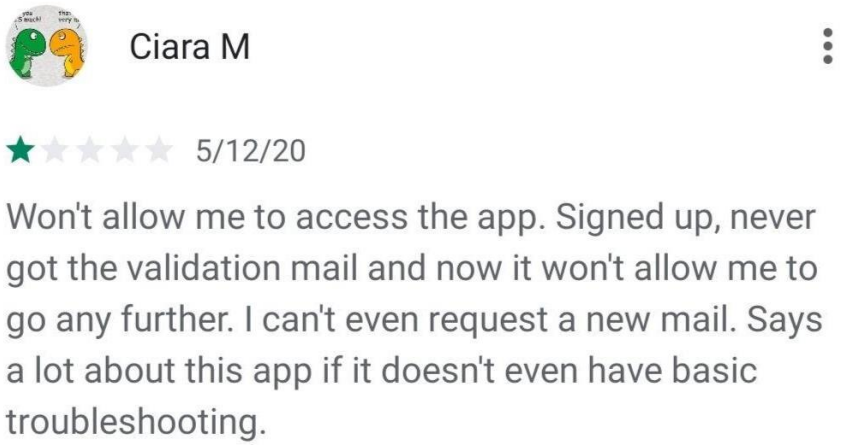

Рисунок 1.4 – Коментар користувача GoodFood

За час користування додатком були відмічені факти спаму інформації з пошти GoodFood без попереднього на те погодження.

Сформуємо список переваг та недоліків сервісу GoodFood.

Переваги:

— механізм формування запиту користувачем;

— підтримка можливості створення власних рецептів;

— сервіс рецептів;

— сторінка пропонованих рецептів;

— збереження вподобаних рецептів.

Недоліки:

— кастомізація інтерфейсу;

— перенасиченість контентом;

— обов’язкова реєстрація;

— час введення даних запиту;

— стабільність роботи додатку;

— присутність реклами.

1.3.3 SideChef

Додаток SideChef має чудово продуманий UI/UX інтерфейс, що дозволяє отримати гарне враження користувачу, який вперше запустив додаток, логічність компонентів введення та виведення інформації та оригінальний підхід до дизайну надають приємний досвід користування програмою, але лише на пристроях із достатньою кількістю оперативної пам'яті та швидкодійним процесором, пристрої із застарілим апаратним забезпеченням страждають від помилок та проблем із заморозкою інтерфейсу користувача [5] (рис. 1.5).

Рисунок 1.5 – Коментар користувача SideChef

Сервіс, що надає послуги формування списку рецептів за запитом користувача не має достатньої локалізації та загалом зберігає порівняно невелику кількість рецептів та фільтрів, за якими їх можна отримувати.

Обов'язкова реєстрація користувача за допомогою Facebook та Google Account, без можливості попереднього ознайомлення з сервісом SideChef, як і в попередніх рішення є недоліком розробки.

Внаслідок не нативності програми можлива нестабільна робота, необхідна більша кількість вільної пам’яті для встановлення. Основним недоліком рішення, що базується на не нативній реалізації є його енерговитратність порівнянно з додатками створеними на базі Android SDK та унеможливлення її покращення з подальшими оновленнями операційної системи (рис. 1.6).

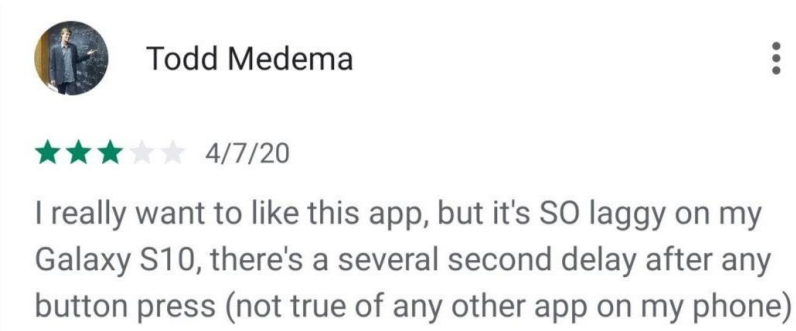

Рисунок 1.6 – Коментар користувача SideChef

Сформуємо список переваг та недоліків сервісу SideChef.

Переваги:

— кастомізація інтерфейсу;

— сторінка пропонованих рецептів;

— сторінка планувальника покупок;

— логічність розміщення компонентів UI.

Недоліки:

— відсутність можливості збереження рецептів;

— сервіс рецептів;

— обов’язкова реєстрація;

— стабільність роботи;

— вага додатку;

— час введення даних запиту.

1.4 Висновки до розділу

На основі повного та структурного аналізу існуючих рішень на платформі Play Store де відбувається розповсюдження програм, що працюють під управлінням операційної системи Android, коментарів людей, що протягом тривалого часу користувалися додатками, були отримані результати успішності популярних проєктів та сформований перелік загальних характеристик, що властиві успішним додаткам.

Більшість позитивних відгуків користувачів відмічають можливості варіативного запиту на сервіс рецептів, підтримку функціоналу по створенню та збереженню власних рецептів на сервісі, зручність розміщення компонентів інтерфейсу та сучасність дизайнерських рішень, можливість збереження минулих запитів та їх результатів на пристрої для використання при відсутності інтернет з'єднання.

Серед недоліків розробок можна відмітити деякі проблеми зі стабільністю додатків, що виражаються у формі ANR, помилок при непередбаченій розробником формі запиту, збою на сервері та внутрішніх помилках програми, відсутність локалізації інтерфейсу і форм запиту користувачем, обов'язковість реєстрації без можливості попереднього ознайомлення з функціоналом додатку, обмеженість фільтрів запиту та недостатній розмір бази рецептів та інгредієнтів але найбільшим недоопрацюванням є зручність формування запиту користувача та час, який витрачається на введення даних.

Основні переваги та недоліки рішень, що не перший рік існують на ринку мобільних додатків, були відмічені під час тестів та аналізу коментарів користувачів, які мають досвід тривалого використання програми були відібрані та сформовані у вигляді таблиці 1.1

Таблиця 1.1 Переваги та недоліки існуючих рішень

Назва	Дизайн	Стабільність	Обов'язкова реєстрація	Сервіс рецептів	Зручність формування запиту
Tasty	+	+	+	+	-
BBC Good Food	-	-	+	+	-
SideChef	+	-	+	-	-

Завдяки отриманій шляхом аналізу інформації поданій в таблиці був сформований перелік основних недоліків та переваг, що властиві конкурентним додаткам. Стабільність і зручність формування запиту користувачем, як основні властиві недоліки, що були відмічені при тестах конкурентних розробок, доповнили основні вимоги до розроблюваного рішення.

Має бути обраний кращий існуючий сервіс для розпізнавання інгредієнтів, щоб користувач отримав максимальну автономність від формування та надсилання запиту для отримання списку рецептів.

Відмічена необхідність обрання кращого сервісу, що пропонує пошук рецептів за інгредієнтами, на основі коментарів користувачів, сформовані вимоги, яким має відповідати сервіс.

Дизайн розробки має відповідати вимогам по ергономічності та наповненню, щоб нівелювати сильні сторони деяких конкурентних розробок.

2 ПРОЄКТУВАННЯ ТА АРХІТЕКТУРА ВЛАСНОГО РІШЕННЯ

2.1 Опис проєкту та вимог

Першим чином для реалізації додатку потрібно обрати та організувати базис архітектури з можливістю подальшого розширення та масштабування програми, бажано мати змогу використовувати спільний для всіх платформ код бізнес логіки (code sharing) незалежно від платформи, на основі якої реалізується представлення. Необхідно передбачити можливість конвертації різного роду графічних форматів до уніфікованого вигляду, що не залежить від операційної системи, підтримуваних форматів і т.д. для стандартизації запиту по розпізнаванню. Наступним кроком є створення специфікацій, за допомогою яких буде відбуватися взаємодія з внутрішніми або зовнішніми API. Основна вимога до сервісу розпізнавання – підтримка достатньої кількості розпізнаваних продуктів та стабільність взаємодії з клієнтською частиною.

Наступним кроком після розпізнавання – є отримання списку розпізнаних інгредієнтів та відправка їх на наступний сервіс, що надає змогу отримання рецептів за обмеженим переліком продуктів. Вся інформація, що отримується, оброблюється та врешті виводиться додатком, має бути відображена у вигляді зручного та логічного інтерфейсу, що задовольняє базовим принципам UI/UX дизайну. Останнім кроком є обрання бази даних та системи управління базами даних(СУБД) для рецептів, що вподобав користувач, формування структури таблиць, та забезпечення коректної роботи принципів транзакцій в базу даних(ACID принципи).

Додаток мінімізує кількість дій, що має зробити користувач для тримання бажаних результатів, повний перелік дій, що здійснює користувач протягом взаємодії з програмою зображений на кресленику ІТ61.190БАК.004 Д4. Додаток функціонуватиме за власним підходом реалізації взаємодії користувача-додатка під час запиту, який базується на мінімалізації кількості часовитратних дій, що

мають привести користувача до бажаного результату(отримання списку рецептів по існуючим інгредієнтам), зображений на рисунку 2.1.

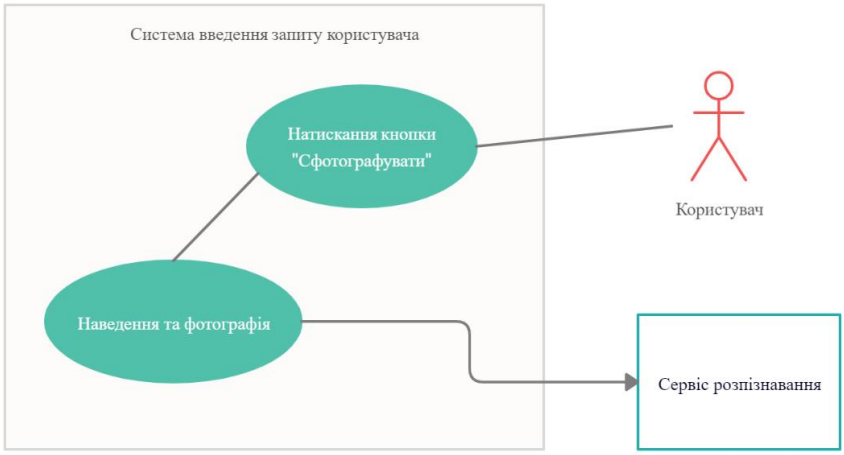

Рисунок 2.1 – Система введення запиту користувачем

Передбачити можливість роботи з додатком у режимі офлайн(перегляд та маніпуляції зі збереженими рецептами).

2.2 Розробка архітектурного рішення

Першим чином для реалізації додатку потрібно обрати та організувати базис архітектури з можливістю подальшого розширення та масштабування програми, бажано мати змогу використовувати спільний для всіх платформ код бізнес логіки (code sharing) незалежно від платформи, на основі якої реалізується представлення

Для задоволення вимог, було розглянуто низку архітектурних рішень, таких як: Гексагональна архітектура, DCI, BCE [17-19]. Перераховані архітектури мають низку спільних підходів, що попри деякі відмінності забезпечують можливість – розподілу завдань між програмними компонентами. Досягається ця мета шляхом розподілу програмного забезпечення на рівні.

Кожна архітектура має не менше одного рівня бізнес-правил і ще один для системного інтерфейсу та інтерфейсу користувача.

Об’єднанням кожного з перелікованих рішень є “Чиста архітектура”, що презентована Робертом Мартіном [16] (рис. 2.2).

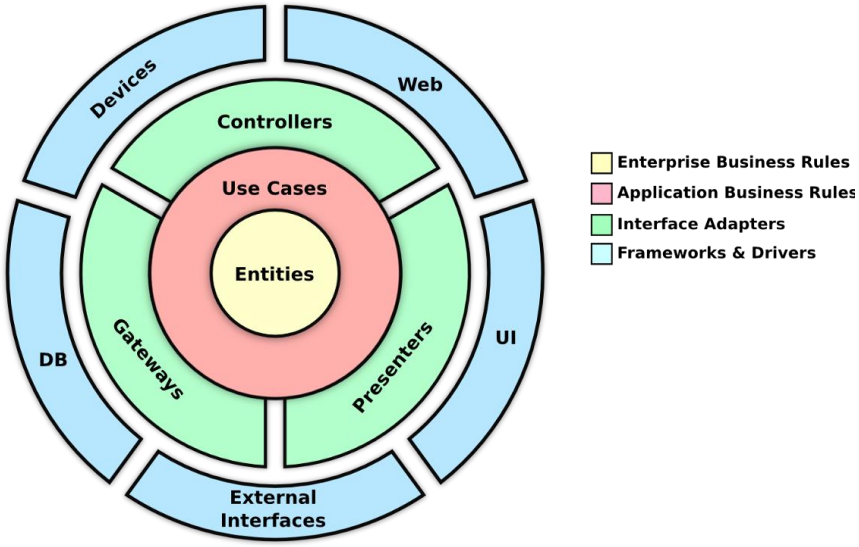

Рисунок 2.2 – Схема архітектурного шаблону “Clean Architecture” [28]

Entities (з англ. сутності) містять у собі критичні бізнес правила рівня розробки. Сутність може бути об’єктом із методами або набором структур даних і функцій. Сутність інкапсулює найзагальніші та найбільш високорівневі правила. Класи сутностей мають бути незалежними від змін в зовнішніх шарах абстракції. Рівень сутностей не має змінюватися під впливом будь-яких програм, що на основі їх побудовані, адже вони є загальними та можуть використовуватися іншими рішеннями [27].

Use Cases (з англ. варіанти використання) містять у собі бізнес правила, що є характерними виключно для розроблюваного застосунку. Вони інкапсулюють і реалізують всі варіанти використання системи. Use Cases організовують потік даних у сутності та з них і вимагають від цих сутностей використовувати їхні критичні бізнес-правила для досягнення своїх цілей [27].

Зміни в зовнішніх рівнях розроблюваного додатку не мають впливати на реалізацію варіантів використання, це допомагає реалізувати один з основних підходів “Clean Architecture” – правило залежності, наприклад зміни в базі даних, інтерфейсі користувача або будь-якому іншому компоненті, що знаходиться на більш високому рівні, не повинні впливати на цей рівень. Рівень варіантів використання ізольований від таких проблем.

Натомість зміни в принципі роботи застосунку безумовно вплинуть на варіанти використання і відповідно, на програмне забезпечення, яке перебуває на цьому рівні.

Interface Adapters (з англ. адаптери інтерфейсу) – це набір адаптерів, що виступають в ролі прошарку між варіантами використання та зовнішніми фреймворками, використовуються для перетворення сутностей, що є зручними в варіантах використання на сутності, що використовуються на зовнішньому рівні та навпаки. Саме на цьому рівні повністю повністю знаходиться архітектура представлення користувача. Презентатори, представлення і контролери – усі належать до рівня адаптерів інтерфейсів. Моделі – часто лише структури даних, які передаються з контролерів до варіантів використання і потім назад із варіантів використання до презентатора та представлення [27].

Аналогічно на цьому рівні перетворюються дані з формату, найбільш зручного для варіантів використання і сутностей, у формат, найбільш зручний для інфраструктури зберігання даних. Ніякий код, що міститься в інших внутрішніх колах, не повинен нічого знати про базу даних. Якщо дані зберігаються в базі даних SQL, тоді весь код мовою SQL повинен бути саме на цьому рівні, точніше, в елементах цього рівня, пов’язаних із базою даних.

Також на цьому рівні містяться будь-які інші адаптери, необхідні для перетворення даних із зовнішнього формату, наприклад отриманих від зовнішньої служби, у внутрішній, який використовують варіанти використання і сутності.

Frameworks & Drivers (з англ. фреймворки та драйвери) є зовнішнім рівнем моделі, що зазвичай складається з таких фреймворків та інструментів, як база даних і вебфреймворк. Зазвичай для цього рівня доводиться писати не дуже багато коду, і дуже часто цей код відіграє роль сполучної ланки. На рівні фреймворків і драйверів зосереджені всі деталі [27].

Між кордонами шарів дані передаються у вигляді простих структур. За наявності бажання можна використовувати найпростіші структури або об'єкти передачі даних DTO. Дані можна також передавати як виклики функцій через аргументи. Або упаковувати їх в асоціативні масиви або об'єкти. Важливо, щоб між шарами передавалися лише прості, ізольовані структури даних. Не треба передавати об'єкти сутностей або записи з бази даних. Структури даних не повинні порушувати правило залежностей [27].

Наприклад, багато фреймворків для роботи з базами даних повертають відповіді на запити в зручному форматі. Їх можна назвати "представленням записів". Такі представлення записів не потрібно передавати через кордони всередину. Це порушує правило залежностей, тому що змушує внутрішнє коло знати щось про зовнішнє [27].

Отже при передачі через кордон між шарами, дані завжди повинні набувати форми, найбільш зручної для внутрішнього кола.

Запропонований архітектурний підхід має наступні характеристики:

— незалежність від фреймворків. Архітектура не залежить від наявності будь-якої бібліотеки. Це дозволяє розглядати фреймворки як інструменти, замість того щоб намагатися втиснути систему в їхні рамки;

— простота тестування. Бізнес-правила можна тестувати без інтерфейсу користувача, бази даних і будь-яких інших зовнішніх елементів;

— незалежність від інтерфейсу користувача. Інтерфейс можна легко замінювати, не змінюючи інші елементи системи;

— незалежність від бази даних. Заміна сховища даних є тривіальною задачею, що не зачіпає бізнес-логіку;

— незалежність від будь-яких зовнішніх агентів. Бізнес-правила не мають посилання на інтерфейси, що ведуть до зовнішнього світу.

Основним принципом даної архітектури є правило залежності, що наголошує на необхідності однонаправленої залежності від зовнішнього кола до внутрішнього.

Отже реалізовувати додаток було вирішено на основі принципів “Clean Architecture” – що є стандартом архітектурного підходу для програм, що базуються на Android SDK та серед вимог яких є тестованість, відділення фреймворкозалежної реалізації в окремий модуль(для можливості підтримки мультиплатформеної парадигми) для інших модулів, підтримка можливості перенесення та оперативної інтеграції модулів бізнес логіки або будь-яких інших між системами (рис. 2.3).

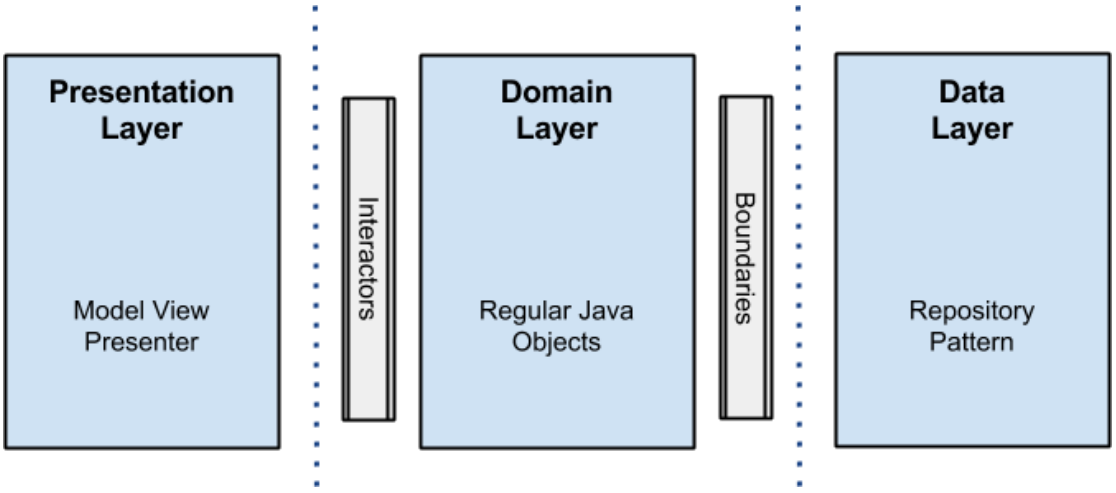

Рисунок 2.3 – Схема архітектурного шаблону “Clean Architecture” адаптованого під Android [29]

Шари бізнес логіки(Domain Layer) та даних(Data Layer) утворюють модулі, які незалежать від платформи на якій вони працюють. Завдяки цьому була використана підтримка мультиплатформеної збірки програм на мові

програмування Kotlin, як зображено на кресленику ІТ61.190БАК.004 Д2, що дозволяє застосувати можливість code sharing для бізнес логіки і специфікацій між платформами і в подальшому адаптувати додаток і для інших операційних систем [6].

Для подальшого розбиття на логічні модулі було прийняте рішення про організацію шару представлення у вигляді шаблону MVP [7], що в свою чергу розподіляє код, що є фреймворкозалежним на додаткові шари, View - існує для відображення результатів роботи бізнес логіки за допомогою SDK інструментів, не має містити в собі бізнес логіку. Presenter – працює лише з бізнес логікою і не має доступу до компонентів SDK, отримує сповіщення про події у View, реагує на них, передає дані у View в обробленому вигляді, делегує поведінку. Model – моделі та класи конвертації, що перетворюють моделі інших шарів у внутрішні моделі, використовуються у Presenter для адаптації даних в зручний для View вигляд (рис. 2.4).

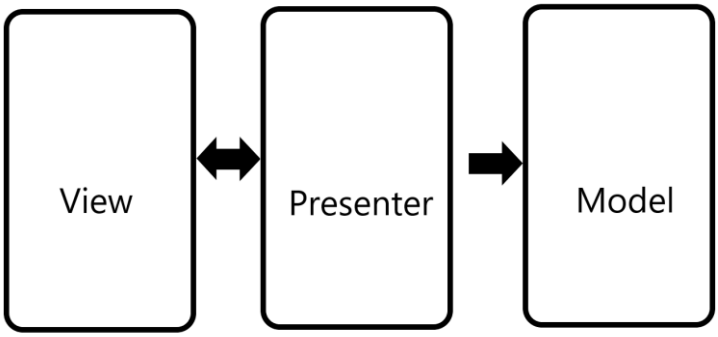

Рисунок 2.4 – Схема архітектурного шаблону MVP

Для реалізації стандартної поведінки компонентів MVP була використана бібліотека Moxy, що кодогенерує поведінку декларовану за допомогою анотацій. Окрім стандартних для архітектурного шаблону MVP класів, бібліотека містить клас ViewState, що забезпечує можливість збереження стану користувача під час

таких операцій як зміна орієнтації екрану, вихід зі сплячого режиму, шляхом збереження переліку викликаних методів в API View та Presenter. ViewState виступаючи в ролі адаптера, при необхідності, дозволяє підключати до одного Presenter декілька View, завдяки чому економляться ресурси пристрою [8] (рис. 2.5).

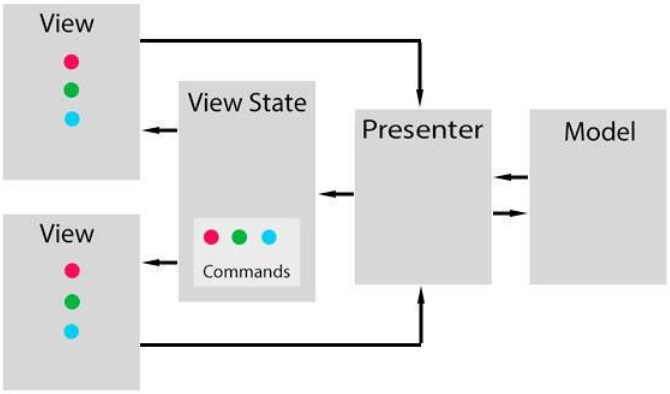

Рисунок 2.5 – Схема архітектурного шаблону бібліотеки Moxy [30]

Для подальшого впровадження внутрішнього та зовнішнього API в шарі даних були створені специфікації, майбутні реалізації котрих мають перевизначати та імплементувати методи взаємодії.

2.3 Висновки до розділу

В даному розділі загальним чином був спроєктований додаток. Після загального аналізу існуючих підходів до побудови архітектури сучасного Android додатку, був погоджений варіант з “Clean Architecture” для всієї розробки, та MVP архітектура для шару представлення, в якому знаходиться вся фреймворко залежна логіка розробки. Описані загальні вимоги до проєкту та система, за якою він взаємодіятиме з користувачем.

3 ВИБІР ДОПОМІЖНИХ СЕРВІСІВ

3.1 Вибір сервісу розпізнавання інгредієнтів

В ході аналізу існуючих рішень для розпізнавання та класифікації об’єктів на фотографії, першим чином було вирішено відмовитися від локального представлення моделі нейронної мережі, через непомірно велику вагу вихідного додатку.

Комп’ютерне бачення – теорія та технологія створення машин, які можуть проводити виявлення, стеження та визначення об'єктів, а також одна із багатьох послуг пов’язаних з нейронними мережами чи нечіткою логікою, що надається сучасними MLaaS платформами, такими як: Amazon Machine Learning services, Azure Machine Learning, Google Cloud AI, IBM Watson за помірну ціну. Технологія базується на Згорткових нейронних мережах(ЗНМ) – клас глибинних нейронних мереж прямого поширення. Рішення базується на принципах багатошарових перцептронів, що за допомогою ядер згортки зменшуються на кожному шарі, при цьому зберігаючи необхідні для подальшої класифікації характерні ознаки попереднього шару (рис. 3.1).

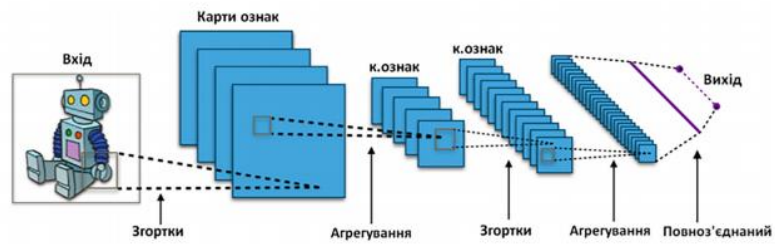

Рисунок 3.1 – Архітектура згорткової нейронної мережі [31]

ЗНМ складається з наступних шарів: шар входу, внутрішні шари, шар виходу. Внутрішні шари ЗНМ складаються зі згорткових шарів, агрегувальних шарів, повноз'єднаних шарів та шарів нормалізації [26].

Згорткові шари мережі об’єднуються у карти ознак, кожен окремий елемент карти – матриця згортування, яка відповідає за окрему ознаку. Операція згортки імітує реакцію кожного окремого нейрону на так званий зоровий стимул, що знаходиться в області бачення рецептивного поля, яке пов’язане лише з одним нейроном (рис. 3.2).

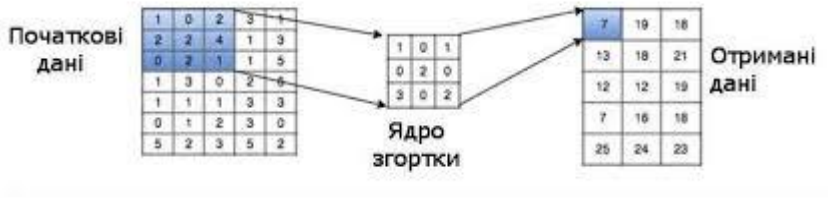

Рисунок 3.2 – Процес згортки [31]

Саме завдяки можливостям зменшення кількості повнозв’язних нейронів без втрати здатності до навчання, ЗНВ отримали практичне значення в теорії штучного інтелекту і обійшли повноз’єднані нейронні мережі прямого поширення в сфері комп’ютерного зору, для їх використання було би необхідним дуже велике число нейронів, навіть у поверхневій (протилежній до глибинної) архітектурі, через дуже великі розміри входу, пов'язані з зображеннями, де кожен піксель є відповідною змінною. Операція згортки дає змогу розв'язати цю проблему, а також проблему зникнення градієнтів при тренуванні звичайних багатошарових нейронних мереж з багатьма шарами перцептронів за допомогою зворотного поширення помилки, оскільки вона зменшує кількість вільних параметрів, дозволяючи мережі бути глибшою за меншої кількості параметрів.

Для оптимізації пошуку предметів на фото незалежно від їхнього розміру, використовуються шари глобального чи локального агрегування, вони допомагають об’єднати поля попереднього шару з нейроном наступного.

Розрізняють 2 види агрегування, усереднювальне та максимізайційне.

В першому випадку, використовується усереднене значення по кластеру, яке й передається наступному шару. В другому варіанті, з кожного кластеру попереднього шару нейронів обирається максимальне значення та передається агрегаційному шару (рис. 3.3).

Рисунок 3.3 – Процес агрегації [31]

Повноз'єднані шари з'єднують кожен нейрон одного шару з кожним нейроном наступного шару. Принцип, що є основою стандартних нейронних мереж прямого поширення – багатошарових перцептронів.

ЗНМ використовують спільні матриці згортки в згорткових шарах, тобто для кожного рецептивного поля шару використовується один і той же фільтр. Завдяки такому підходу оптимізується обсяг пам'яті, що необхідна для збереження моделі нейронної мережі, а також поліпшує продуктивність її роботи.

Так як у відкритому доступі відсутні моделі нейронних мереж, що навчені класифікувати лише продукти харчування, доведеться використовувати сервіси, що мають найбільшу кількість розпізнаваних об’єктів. Далі розглянемо варіанти віддалених сервісів, що надають послуги по розпізнаванню об’єктів на фотографії. Порівняємо їх за варіативністю типів об’єктів, які ці моделі можуть розпізнати та ціною за використання послуг сервісів.

Cloud Vision API – це сервіс, що працює виключно на розпізнавання зображень, розробляється компанією Google та є одним з найпопулярніших

рішень при необхідності класифікації, секторизації об’єктів, розпізнаванню тексту [9].

Можливі шляхи використання:

— маркування об’єктів;

— знаходження облич та визначення емоцій, але без ідентифікації людини;

— опис процесів на фото або знаходження пам’яток;

— знаходження тексту по фотографії та ідентифікація мови, на якій він написаний;

— знаходження переважаючого кольору.

Сервіс також підтримує тренування власних моделей нейронних мереж [9].

Ціноутворення за послуги Cloud Vision API цілком залежить від кількості сервісів, які одночасно використовуються для задоволення запиту користувача. Перші 1000 запитів в місяць є безкоштовними, кожен наступний запит по 5 000 000 запит включно, ціна варіюється від $1.50 до $3.50 в залежності від сервісів, що необхідні. Якщо кількість запитів на місяць буде в проміжку 5 000 001 – 20 000 000, маркування кожного зображення коштуватиме $1.

Наступним можливим рішенням є використання Microsoft Azure Cognitive Services, розробляються компанією Microsoft, сервіси комп’ютерного бачення, що входять в цей пакет класифіковані на 6 груп, що виконують аналіз відео та зображень, розпізнавання облич, розпізнавання тексту [23].

Ознайомимося з повним переліком можливостей сервісу:

— розпізнавання та маркування об’єктів, дій;

— розпізнавання написаного чи надрукованого тексту;

— знаходження переважаючого кольору;

— модерація контенту;

— розпізнавання облич, емоцій, груп людей, вік;

— наявний застосунок, що допомагає знаходити людей у відео, визначати настрій розмови.

Ціна використання сервісу залежить від багатьох чинників, таких як, сервіс, зо використовується, регіон розробника, кількість зворотніх транзакцій, що незалежать від кількості запитів. Наприклад для регіону Східної Європи кількість запитів в проміжку 0 - 1 000 000 коштуватимуть $1.50 за кожну 1000 транзакцій, у відрізку від 1 000 000 до 5 000 000 транзакцій, кожна 1000 коштуватиме $1, всі наступні запити здійснюватимуться за ціною $0.65.

Amazon Rekognition API – сервіс, що підтримується компанією Amazon дозволяє влаштовувати аналізатор відео та зображень [24]. Базується на тих самих технологіях аналізу, що використовуються в сервісі Amazon Photos.

Можливі шляхи використання:

— розпізнавання об’єктів на фото та їх класифікація;

— на відео сервіс розпізнає види діяльності;

— розпізнавання та порівняння облич;

— аналізатор емоцій;

— модерація контенту на відео;

— розпізнавання знаменитостей на відео або зображенні.

Сервіс також підтримує тренування власних моделей нейронних мереж.

Ціна використання API сервісу варіюється в залежності від регіону, кількості сервісів, що використовуються. Перші 12 місяців дозволено безкоштовно аналізувати 5000 зображень у місяць. В подальшому, 1 000 000 зображень коштуватиме $1.20 за кожні 1000 аналізів.

Наступні 9 000 000 - $0.96 за кожні 1000 зображень. В подальшому, до 90 000 000 аналізів включно за кожну 1000 доведеться викласти $0.72.

Якщо кількість запитів перевищить 100 000 000, кожна 1000 буде коштувати $0.48.

Watson Visual Recognition – сервіс, що розробляється компанією IBM, на даний момент не підтримує аналіз відео, але є аналіз зображень, за допомогою 4 великих моделей нейронних мереж, що підтримують:

— розпізнавання об’єктів;

— розпізнавання обличчя, віку та статі людини;

— розпізнавання їжі на фото;

— модерація контенту;

— розпізнавання тексту (знаходиться в бета тестуванні) [25].

Сервіс також підтримує тренування власних моделей нейронних мереж.

Варто відмітити, що IBM, що модель нейонної мережі для розпізнавання їжі на фото розробляється компанією окремо.

Ціноутворення базується на двох цінових планах. Перший, легкий, що дозволяє користувачам безкоштовно аналізувати до 1000 зображеня на місяць, створювати і тренувати до двух моделей власних нейронних мереж. Стандартний план включає розпізнавання з власних та сервісних моделей за $0.002 за кожне зображення, та тренування власної моделі за $0.10 кожного місяця.

Беручи до уваги результати аналізу існуючих MLaaS сервісів, що пропонують розпізнавання та класифікацію об’єктів на зображеннях, сформуємо таблицю, де наочно зобразимо переваги та недоліки кожного сервісу, оберемо найкращий, базуючись на таблиці 3.1 та пропонованих цінових планах сервісів.

Таблиця 3.1 Порівняння послуг MLaaS сервісів

Сервіс	Amazon Rekognition API	Microsoft Azure Cognitive Services	Cloud Vision API	Watson Visual Recognition
Розпізнавання об’єктів	+	+	+	+

Сервіс	Amazon Rekognition API	Microsoft Azure Cognitive Services	Cloud Vision API	Watson Visual Recognition
Розпізнавання сцен	+	+	+	-
Розпізнавання облич	+	+	+	+
Модерація контенту	+	+	+	+
Розпізнавання тексту	+	+	+	+
Пошук схожих зображень	-	-	+	-
Знаходження місць	-	+	+	-
Розпізнавання їжі	-	-	-	+
Розпізнавання основного кольору	-	+	+	-

Виходячи з результатів створеної таблиці, можливо припустити, що сервіс Watson Visual Recognition, який має в своєму розпорядженні модель нейронної мережі, що навчена розпізнавати лише продукти харчування, найкраще підходить для розпізнавання інгредієнтів розроблюваним рішенням, але судячи з тестів сервісів на розпізнавання продуктів, явним фаворитом, по розпізнаванню

інгредієнтів є сервіс Cloud Vision API, який складається з багатьох нейронних мереж та розпізнає інгредієнти точніше, аніж Watson Visual Recognition. Порівняємо ціноутворення на послуги різних сервісів в таблиці 3.2.

Таблиця 3.2 Порівняння цін на послуги MLaaS сервісів

Запитів на місяць	Amazon Rekognition API	Microsoft Azure Cognitive Services	Cloud Vision API	Watson Visual Recognition
1000	free	free	free	free
5 000 000	$0.00096	$0.0015	$0.0015	$0.002
20 000 000	$0.00072	$0.001	$0.001	$0.002
100 000 000	$0.00048	$0.00065	$0.001	$0.002

Кращим сервісом для розпізнавання інгредієнтів на зображені виявився Cloud Vision API, що має найбільшу кількість розпізнаваних об’єктів, включаючи їжу, підтримує можливість подальшої міграції з послуг готової моделі, до власної, яку можна так само тренувати в кабінеті сервісу, ціна хоча і не є найнижчою серед конкурентів, але сервіс є найкращим серед конкурентів для розпізнавання об’єктів на зображенні. При тому, що використання сервісу при кількості запитів > 50 000 на місяць є недоцільним, по досягненню цієї цифри необхідно мати власну діючу модель нейронної мережі.

3.2 Вибір сервісу пошуку рецептів

На сьогоднішній день існує різноманітна кількість сервісів, основна функція яких фільтрування та видача інформації за багатьма категоріями: харчування, продукти, ресторани, рецепти, напої, дієти, органіка та багато

інших. Наступною задачею є обрання сервісу, що має API з пошуком рецептів за обмеженою кількістю інгредієнтів, сервіс має містити якомога більшу базу рецептів та якомога меншу ціну за запит. Для обрання кращого варіанту було вирішено сфокусуватися на найпопулярніших сервісах, що надають необхідні послуги.

Edamam API – один з найбільших сервісів, що надає можливості пошуку рецептів за інгредієнтами, отримання інструкцій та фотографій самої страви. Для отримання доступу до рецептів та авторизації запитів використовується ключ користувача, що створюється в кабінеті сервісу Edamam API [21]. Запити виконуються на основі безпечного протоколу HTTPS, що шифрує передавані користувачем дані. API сервісу підтримує стиснення отримуваних данних, що при великих об’ємах запитів може зекономити велику кількість трафіку. Сервіс підтримує 2 мови – англійську та іспанську. Сховище рецептів зберігає більше 1 500 000 рецептів англійською мовою та 200 000 рецептів

Відповідь, що приходить на запит, отримується в форматі JSON або HTML сторінки, у випадку, якщо на сервері відбулася помилка.

Ціноутворення в Edamam поділяється на 3 групи: Developer, Startup, Enterprise, що відрізняються кількістю дозволених запитів на хвилину з одного облікового запису, кількістю запитів на місяць, кількістю підтримуваних фільтрів та відповідно сумою оплати.

Користувач з тарифним планом Developer, що розповсюджується безкоштовно, отримає 5000 запитів на місяць, що обмежуються 5 запитами на хвилину, варіацію з 10 фільтрів, що відсіюють алергічні продукти, дієти. Відповідь із серверу обмежена 100 одночасних результатів.

Startup – тариф, що також надається безкоштовно, але при безпосередньому запиті до служби підтримки розробником, надає до 20 000 запитів на місяць, обмежених 15 запитами в хвилину, підтримує 30 дієтичних та алергічних фільтрів, та як і попередній варіант надає до 100 результатів на запит.

Enterprise – рішення для розробок, що впроваджуються або вже діють на ринку з великою кількістю користувачів. Оплата здійснюється щомісяця в розмірі $29 для кількості запитів, що не переважають 50 000, кожен наступний запит коштуватиме розробнику $0.01, основною перевагою перед іншими тарифними планами, крім сумарної кількості запитів, що надаються на місяць є необмежена кількість запитів на хвилину, підтримується більше 30 фільтрів та до 1000 одночасних результатів [21].

Сформуємо таблицю 3.3 на основі наданої інформації.

Таблиця 3.3 Порівняння тарифних планів Edamam

	Developer	Startup	Enterprise
Кількість запитів / місяць	5000	20000	50000
Кількість запитів / хвилина	5	15	29
Кількість фільтрів	10	30	30
Кількість результатів	100	100	1000
Ціна / місяць	Free	Free	$29
Ціна подальших запитів / запит	-	-	$0.01

Spoonacular API – сервіс, що підтримується однойменною компанією та пропонує розробникам доступ до 360 000 рецептів та 80 000 продуктів харчування [10]. Система сервісу дозволяє здійснювати пошук природньою мовою. Наприклад: “Gluten free apple cake”. Користувачі API мають можливість візуалізації списку інгредієнтів, аналізу ціни інгредієнтів, фільтрацію по

улюбленим продуктам, пошук по класифікаціям рецептів, конвертації ваги в кількість інгредієнтів та вирахування плану харчування. Основною перевагою над попереднім сервісом є можливість пошуку рецептів в режимі “Що в холодильнику”, який дозволяє обмежити результуючий список рецептів за переліком наявних інгредієнтів. Сервіс підтримує пошук англійською мовою [10].

Spoonacular API має в своєму розпорядженні 5 тарифних плани, що розрізняються доступною кількістю запитів на день, ціною та спеціальними пропозиціями.

Тарифний план Free – є безкоштовним рішенням для розробників, що бажають протестувати сервіс, пропонує 150 запитів на день без можливості отримання додаткових запитів. Дозволяє отримати доступ до форуму, де можна отримати підтримку в разі виникнення питань. Обов’язковим є вказання зворотнього посилання на сайт Spoonacular при використанні цього рішення.

Наступним тарифним планом є Cook, що є першим платним рішенням з ціною в $29 на місяць, відкриває доступ до 1 500 запитів на день та можливість подальшого розширення тарифного плану за $0.005 при кожному наступному запиті. Відкрита підтримка розробника за допомогою пошти. В цьому та наступних рішення необов’язково вказувати посилання на сервіс Spoonacular API.

Culinarian – платне рішення, що коштуватиме $79 на місяць з можливістю 4 500 запитів на день з подальшим розширенням, $0.004 за кожен наступний запит, як і попередній тариф, реалізує підтримку по пошті.

Chef – наступний тариф, що коштуватиме $149 на місяць, відкриває можливість запиту підтримки по телефону, доступ до ексклюзивних оновлень API, 10 000 запитів на день, та оплати кожного наступного за $0.002

Enterprise – тарифний план коштуватиме від $300 та дозволяє користувачу API напрямку спілкуватися з розробниками, пропонувати власні рішення, при всіх перевагах попередніх тарифів.

На основі результатів аналізу сервісу, сформуємо таблицю 3.4.

Таблиця 3.4 Порівняння тарифних планів Spoonacular

	Free	Cook	Culinarian	Chef	Enterprise
Кількість запитів / день	150	1500	4500	10000	Безлімітно
Ціна подальших запитів / запит	-	$0.005	$0.004	$0.002	Договірна
Підтримка поштою	-	+	+	+	+
Підтримка телефоном	-	-	-	+	+
Ціна / місяць	$0	$29	$79	$149	> $300

BigOven.com Recipe API – сервіс, що пропонує доступ до більше ніж 500 000 рецептів та супровідних даних, підтримку детальної інформації про рецепти включаючи фото, відео, список інгредієнтів [22]. Основним недоліком сервісу є те, що він надає рецепти лише в вигляді посилань на веб-ресурси, що ускладнює можливість створення зручного та адаптивного додатку.

Ціноутворення сервісу поділяється на 5 рішень: “Basic”, “Bronze”, “Silver”, “Gold”, “Platinum”. Тарифи різняться ціною, кількістю запитів на годину, можливістю отримувати специфічну інформацію по кожному рецепту, таку як кількість калорій, протеїнів, жирів, преміальні версії тарифів підтримують пошук вегетаріанських, безглютенових рецептів. Розглянемо кожне рішення окремо:

Basic – тариф, що дозволяє отримати базову версію API з можливістю пошуку за назвою, інгредієнтами, ключовими словами. Підтримує 500 запитів щогодини, виключає можливість специфічного пошуку і запитів, ціна складає $99 на місяць.

Bronze – на відміну від базового рішення підтримує надання розширеної інформації по рецептам, кількість запитів обмежена 1000 на годину, не підтримує розширеного пошуку, ціна на підписку складає $199 щомісяця.

Silver – тариф, що включає всі можливі форми запиту та розширеної відповіді API, обмежує кількість запитів з одного облікового запису до 5 000 запитів щогодини, ціна підписки складає $399 на місяць.

Gold – тариф, що розповсюджується за ціною в $699 доларів щомісячно, та дозволяє розширити кількість запитів користувачем до 10 000 на годину, як і попередній тариф, включає всі можливі розширення.

Platinum – знімає будь-які обмеження на кількість запитів, розповсюджується виключно за договірною ціною, що залежить від потреб розробника

Сформуємо таблицю 3.5 на основі наданої інформації.

Таблиця 3.5 Порівняння тарифних планів BigOven.com Recipe

	Basic	Bronze	Silver	Gold	Platinum
Кількість запитів / годину	500	1000	5000	10000	Безлімітно
Розширений результат	-	+	+	+	+
Розширений запит	-	-	+	+	+

	Basic	Bronze	Silver	Gold	Platinum
Комерційне використання	+	+	+	+	+
Ціна / місяць	$99	$199	$399	$699	Договірна

За результатом аналізу існуючих на ринку рішень, що пропонують сервіси надання рецептів за фільтрами, було прийняте рішення про використання сервісу Spoonacular API, що має лояльні тарифні плани, велику кількість рецептів, що поставляються не у вигляді вебпосилань, а повноцінними інструкціями, що покращують можливість оптимізації та кастомізації інтерфейсу користувача, крім того, вирішальною перевагою рішення є те, що в ньому присутній фільтр “Що в холодильнику”, який дозволяє обмежити результуючий список рецептів за переліком наявних інгредієнтів. Сервіс BigOven.com Recipe не є оптимальним рішенням по причині відсутності безкоштовного тарифного плану та викристанні вебпосилань замість повноцінних інструкцій для рецептів. Edamam хоча і відповідає всім необхідним критеріям, але пошукова система не є оптимальною для запитів розроблюваного сервісу та має оптимізуватися, але в подальшому при розвитку додатка, можлива імплементація API і цього сервісу.

3.3 Вибір бази даних

Наступним кроком є вибір бази даних, що на даний момент використовується для зберігання вподобаних рецептів. Було вирішено відмовитися від підходу з реєстрацією користувача на сервісі та зберігання його рецептів на сервері. Користувач має отримувати всі можливі функції додатку працюючи з ним навіть в офлайні, що і забезпечує локальна база даних. Сучасні сховища даних поділяються на реляційні та нереляційні рішення, що

відрізняються швидкодією, зручністю та оперативністю внесення змін користувачем, підходом до міграції, відповідністю ACID принципам.

Реляційні бази даних – зберігають дані у вигляді завчасно задекларованих таблиць, що забезпечують строгу типізацію записів, які в них зберігаються. Складаються з атрибутів, кортежів та відношень (рис. 3.4).

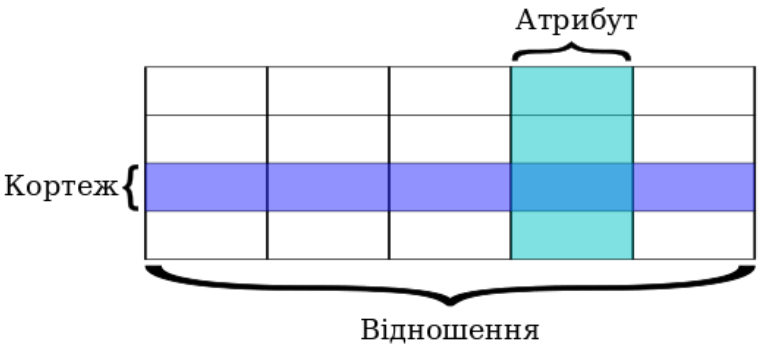

Рисунок 3.4 – Компоненти реляційної бази даних

Реляційна база даних може відповідати різним нормальним формам, від яких залежить ступінь того, наскільки багато надмірної інформації в ній зберігається.

Розрізняють наступні нормальні форми:

а) першу нормальну форму;

б) друга нормальна форма;

в) третя нормальна форма;

г) нормальна форма Бойса — Кодда;

д) четверта нормальна форма;

е) п'ята нормальна форма.

Реляційна база даних відповідає критеріям першої нормальної форми лише у випадках коли кожна таблиця сховища має ключ, за допомогою якого можна ідентифікувати запис, таблиця не має записів, що зустрічаються два або більше

разів, на пересічені кожного атрибута з кожним кортежем, знаходиться лише одне значення.

Сховище відповідає вимогам другої нормальної форми лише коли є відповідність першій нормальній формі та відсутні потенційні ключі, від яких залежать інші атрибути, у випадку якщо такі ключі є, таблиця має бути розбита на додаткові таблиці, що в свою чергу відповідатимуть другій нормальній формі.

База даних відповідає третій нормальній формі лише у випадку відповідності другій нормальній формі та коли всі атрибути, що не є ключем, залежать лише від ключа, але не від інших атрибутів.

Форма Бойса-Кодда є більш суворим представленням третьої нормальної форми, що накладає додаткову вимогу, яка визначає можливість лише однонаправленної залежності, тобто лише прості атрибути можуть залежити від ключа, але ключ не може залежити від простих атрибутів.

Реляційна база даних знаходиться в четвертій нормальній формі лише у випадку, коли вона знаходиться в формі Бойса-Кодда та не має багатозначних залежностей.

Відповідність п’ятій нормальній формі залежить від відповідності четвертій нормальній формі, другою умовою відношення до п’ятої нормальної форми є залежність нетривіальних залежностей цілого потенційного ключа, але не до його частини.

Нереляційні бази даних – це сховища, основною задачею яких є нетривіальність масштабування та атомарність операцій, що виконуються над ними. Основними ознаками нереляційних баз даних є невідповідність ACID принципам, але швидке внесення змін та масштабування. Існують наступні типи таких сховищ:

Ключ – значення, зберігають дані, як і сказано в назві у вигляді відношення ключа - значення, тобто за унікальним ключем можна зберігати та отримувати інформацію, що зберігається (рис. 3.5).

Key	Value
K1	ABC
K2	DEF,FED
K3	3, ZERO,01/01/2015

Рисунок 3.5 – Принцип роботи сховища “Ключ – значення”

Найчастіше використовується, якщо в системі важлива масштабованість. Чудовим прикладом використання такої системи є SharedPreferences в Android, що дозволяють зберігати налаштування користувача у вигляді ключа – значення.

Column Family (з анг. сім’я стовпчиків) – нереляційна база даних, принцип якої в існуванні рядку, ідентифікатор якого і є ключем, в якому зібрані стовпчики зі згрупованими даними (рис. 3.6).

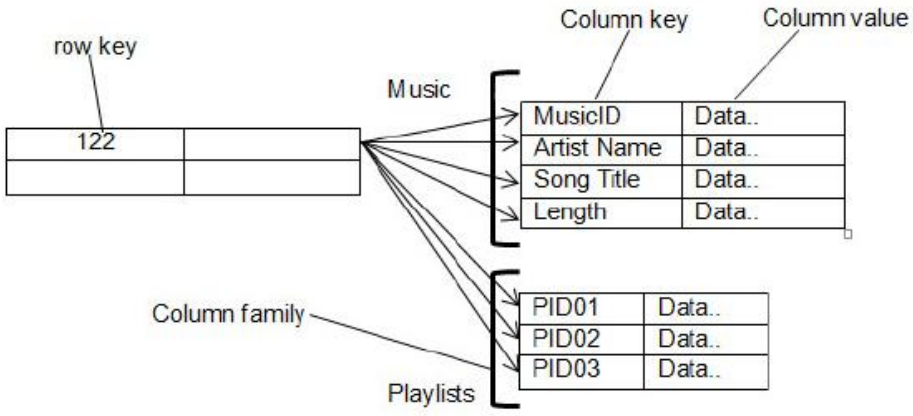

Рисунок 3.6 – Принцип роботи сховища Column Family

Технологія знайшла своє розповсюдження в таких системах, як база даних BigTable, що розробляється та підтримується компанією Google. Такі системи як Cassandra та HBase також базуються на даній технології.

Документоорієнтовані СУБД – забезпечують зберігання даних у вигляді ієрархічної структури, найчастіше дерева. Використовуються у видавничій справі та пошуку документів (рис. 3.7).

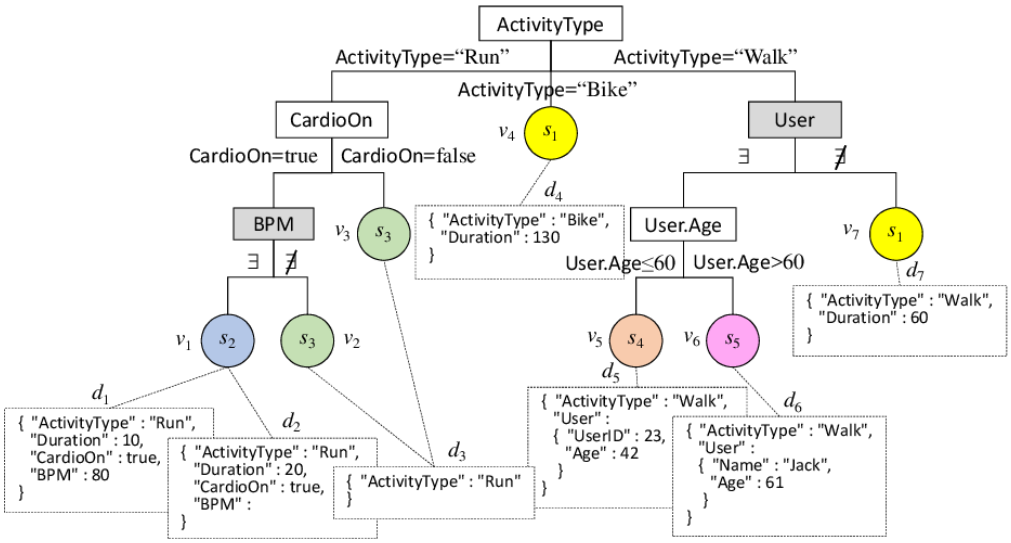

Рисунок 3.7 – Принцип роботи сховища Документоорієнтовані СУБД [35]

Графові СУБД – це система управління графовими базами даних. Основними елементами з яких складається модель графової бази даних є вузли та зв'язки між ними. Графові СУБД орієнтовані для задач пошуку зв'язків в системах з великою кількістю зв'язків, зазвичай такі системи використовуються в сховищах, якщо між записами існує багато зв'язків, зазвичай використовуються в біоінфоматиці, для моделювання сполук, соціальних мережах, щоб прослідковувати знайомства між користувачами та створювати рекомендації користувачу стосовно нових знайомств. При великій кількості зберіємих даних, вся необхідна аналітична робота виконується за допомогою спеціальних механізмів графових обчислень. На відміну від попередніх NoSQL рішень, графові СУБД, як правило, підтримують ACID принципи (рис. 3.8).

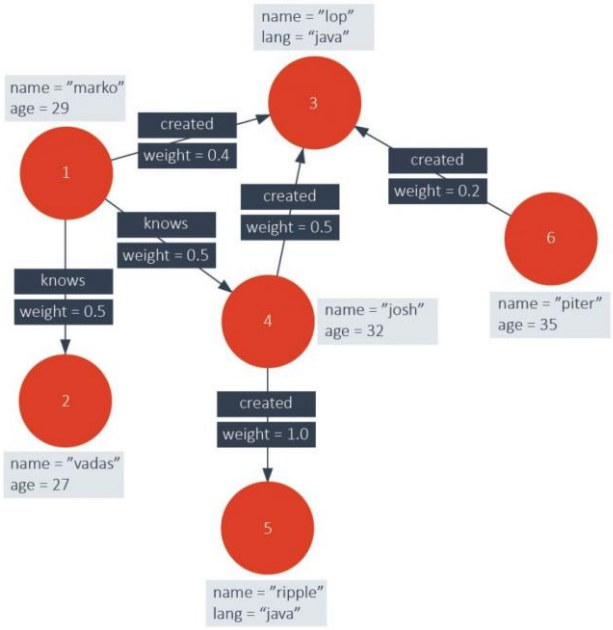

Рисунок 3.8 – Принцип роботи сховища Графової СУБД [34]

Серед основних переваг нереляційних сховищ по відношенню до реляційних є їхня швидкодія, масштабованість. Сховища, що в свою чергу зберігають дані у вигляді реляційної бази даних гарантують дотримання ACID принципів.

Перерахуємо деякі переваги, що отримує додаток при наявності локальної бази даних:

— підтримка роботи в офлайні;

— при наявності кеша, економиться трафік користувача;

— стабільна робота додатку, що є незалежною від наявності інтернет підключення;

— персональні дані можуть зберігатися локально, що є вимогою багатьох користувачів.

Розглянемо варіанти існуючих баз даних, що є портованими на операційну систему Android та є достатньо популярним серед розробників.

Level DB – це нереляційна база даних, що розроблена двома інженерами компанії Google, що заснували свою компанію LevelDB Team. Написана на мові програмування C++. Існує версія для обох мобільних операційних систем iOS та Android. Серед основних переваг розробки над іншими рішеннями є швидкодія деяких операцій та розмір самої бази даних, що мінімізується за допомогою Google's Snappy бібліотеки, яка стискає дані. Записи зберігаються у вигляді ключа-значення, записи відсортовані за ключем. Відсутня можливість синхронізації з сервером, для цього необхідно писати свою обгортку. Для збереження швидкодії, відсутнє шифрування даних, що зберігаються. Користувачами сховища відмічається загальна ненадійність та нестабільна міграція старих версій нас нові. Вага порожньої СУБД залежить від операційної системи, але в середньому не перебільшує 350 KB. Розповсюджується під ліцензією New BSD.

SQLite – СУБД, що працює на основі реляційної бази даних, є вбудованим сховищем для операційних систем iOS та Android. Розроблена компанією Hwaci [11]. Повною мірою забезпечує відповідність принципам ACID за рахунок створення файлу журналу, для запису даних в сховище блокується весь файл бази даних, читання записів може відбуватися з декількох потоків. SQLite підтримує динамічну типізацію. Можливі типи даних, що зберігаються: INTEGER, REAL, TEXT, BLOB, NULL. Проєкт має власну архітектуру, що допомагає краще ознайомитися з рішенням (рис. 3.9).

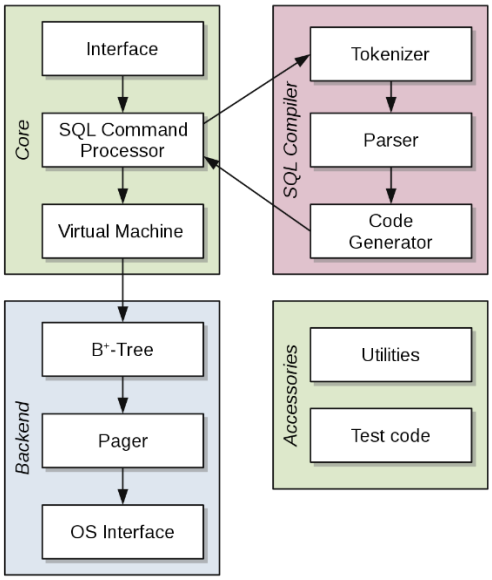

Рисунок 3.9 – Архітектура SQLite [11]

Не підтримується синхронізація з сервером. Не має автоматичного шифрування даних, але підтримує можливість шифрування бази даних за допомогою бібліотеки SQLCipher. Середній розмір порожнього сховища не перебільшує 500 KiB. SQLite є Open Source, тому програмний код може бути модифікований та запропонований співтовариству.

Interbase – реляційна СУБД, що розробляється компанією Embarcadero, існує декілька паралельно розроблюваних версій СУБД, комерційна та Open Source. Висока надійність рішення гарантується тим, що послугами СУБД користується компанія Boeing при розрахунку крил літаків на жорсткість. Основною перевагою обох версій над існуючими конкурентними розробками є використання версійованої архітектури зберігання даних, що забезпечує можливість одночасного читання та редагування записів різними процесорними потоками. В залежності від версій, СУБД підтримує клієнт-серверну архітектуру та можливості шифрування як мережевого трафіку так і таблиць і баз даних користувача на клієнтській машині. Присутня підтримка шифрування даних

SQL Anywhere – реляційна СУБД, що розробляється та розповсюджується під приватною ліцензією компанії Sybase iAnywhere. Пропонує клієнт-серверну архітектуру, що забезпечує можливість синхронізації даних з сервером. Підтримується можливість шифрування сховища за допомогою симетричного алгоритму AES.

Сформуємо таблицю 3.5 баз даних, що підтримуються в Android та яка характеризуватиме кожне окреме рішення [14].

Таблиця 3.5 Порівняння існуючих баз даних

Назва	Level DB	SQLite	Interbase	SQL Anywhere
Підтримка Android / IOS	Android / iOS	Вбудована в Android / iOS	Android / iOS	Android / iOS
Тип зберігання даних	Нереляційний(ключ значення)	Реляційний	Реляційний	Реляційний
Синхронізація з сервером	-	-	Залежить від версії	Залежить від версії
P2P синхронізація	-	-	Залежить від версії	-
Шифрування даних	-	-	Залежить від версії	AES-FIPS
Ліцензія	New BSD	Public domain	Proprietary	Proprietary
Мінімальна вага	350 KB	500KiB	-	-
Компанія розробник	LevelDB Team	Hwaci	Embarcadero	Sybase iAnywhere

На основі структурного аналізу існуючих рішень, можна зробити висновки, що СУБД, які розповсюджуються під приватними ліцензіями з великою кількістю додаткового функціоналу та можливістю підтримки клієнт-серверної архітектури є рішеннями, що на даний момент не є затребуваними. Нереляційна Level DB СУБД, хоча і є достатньо швидкодійною розробкою, не задовольняє вимоги стабільності та безвідмовності. Реляційна СУБД SQLite, що поставляється разом з Android SDK, та не потребує додаткових бібліотек, є на даний момент найкращим рішенням для зберігання структурованих даних на мобільному пристрої. Для більшої абстракції на рівні маніпуляцій записами в базі даних безпосередньо об’єктами використовуватиметься високорівневий інтерфейс Room [12], що працює за допомогою низькорівневих прив’язок до SQLite, також є частиною Android Architecture бібліотек, які підтримуються та просуваються компанією Google для уніфікації стандартів написання Android додатків.

3.4 Висновки до розділу

Результатом дослідницької роботи в цьому розділі є проведений аналіз сервісів, що надають послуги розпізнавання інгредієнтів за фото та надання переліку рецептів за запитом, а також СУБД. За результатами аналізу, найкращим сервісом для розпізнавання є Cloud Vision API, сервіс, що надає перелік рецептів за інгредієнтами - Spoonacular та СУБД SQLite.

4 РЕАЛІЗАЦІЯ ПРЕДСТАВЛЕННЯ ДОДАТКУ ТА ПОРІВНЯННЯ З КОНКУРЕНТАМИ

Найбільш помітною частиною будь-якого додатку для користувача є його інтерфейс. Розробка графічного інтерфейсу користувача це не лише обрання кольорової палітри додатку та вибір шрифту. Кнопки та інші елементи взаємодії користувача мають знаходитися в тих місцях інтерфейсу, де користувач їх шукатиме, тобто їх місцезнаходження має бути логічним для людини, що вперше відвідала додаток. Анімація також є одним зі способів, що може зробити інтерфейс користувача зручним, цікавим та інтерактивним.

4.1 Дизайн

На сьогоднішній день, серед мобільних додатків найбільш розповсюджені два типи UI/UX дизайну Material Design та Human Design.

Material Design – стиль дизайну інтерфейсів користувача, що розробляється компанією Google та пропагується як такий, що може використовуватися на більшості платформ де існує графічна взаємодія програмного забезпечення з користувачем. Перша версія Material Design була представлена на конференції Google I/O 25 червня 2014 року. Повноцінна реалізація дизайну в операційну систему Android відбулася з версії Android Lollipop, всі подальші версії операційної системи та компоненти додатків, що під неї розробляються продовжують оновлюватися до більш сучасних версій Material Design (рис. 4.1)

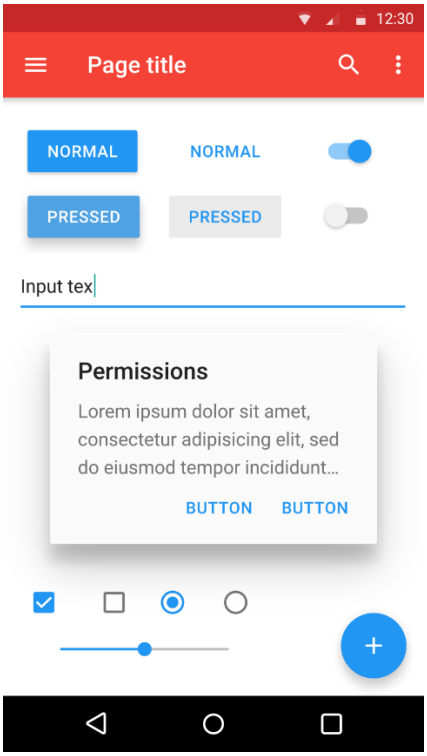

Рисунок 4.1 – Основні компоненти Material Design [32]

Стиль Material Design грунтується на наближенні анімацій, та компонентів до законів, що властиві фізичному світу. Кожен компонент, що знаходиться вище за інший має відкидати тінь. Картки, на основі котрих базується розміщення інформаційних компонентів, зображень, представлення списків мають мати заокруглені кути(рис. 4.2).

Рисунок 4.2 – Дизайн елементу списку на основі Material Design [32]

Анімація, що відображається при зміні екранів, при їх відкритті або при будь-яких інших взаємодіях користувача з інтерфейсом, має бути максимально непомітною та візуально мати фізичні властивості, щоб виглядати більш натурально.

Операційна система Android має стандартний шрифт під назвою Roboto(рис. 4.3).

Рисунок 4.3 – Стандартний шрифт Android [33]

В Material Design існує два підходи до навігації між сторінками в додатку.

Перший відображається у вигляді спеціального меню, яке викликається за допомогою спеціальної кнопки або шляхом витягненням компоненту з краю екрану.

Другий навігаційний компонент відображається в вигляді ряду кнопок в верхній або нижній частині екрану пристрою. Використання кожного підходу, разом із базовою реалізацією анімації при зміні екранів компонентів навігації, створюють враження безпосередньої фізичної взаємодії з графічним інтерфейсом програми[15] (рис. 4.4).

Рисунок 4.4 – Дизайн елементу списку на основі Material Design [32]

Дизайн та анімація натиску кнопок є наближеними до матеріального світу, як і в інших компонентах інтерфейсу. Існує два стилі, що просуваються в Material Design Guidelines, як основні, це плоскі та випуклі (рис. 4.5).

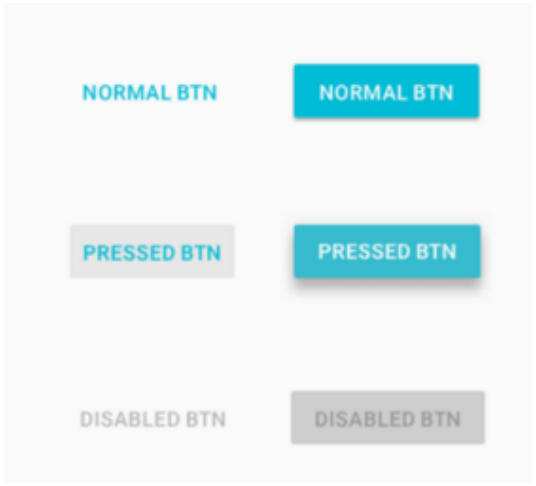

Рисунок 4.5 – Дизайн елементу списку на основі Material Design [32]

Текст, що зображує дію для якої створено кнопку по стандарту пишеться в верхньому регістрі.

Material Design є чудовим прикладом логічного дизайну, навіть для людини, що взаємодіє з ним вперше, саме завдяки наближеності стилів та анімацій до матеріального світу з котрим взаємодіє кожна людина.

Apple Human Interface Guidelines(AHIG) – підхід до дизайну продуктів під управлінням операційних систем розроблених компанією Apple. Вперше впроваджений разом з операційною системою iOS 7. На відміну від попереднього підходу до дизайну інтерфейсів, Основною ідеєю AHIG є просування “повітряності” та легкості графічного інтерфейсу взаємодії з користувачем.

На відміну від Material Design, AHIG не має стандартних візуальних компонентів навігації між основними розділами додатку, розробник має використовувати панель вкладок, що виконує функцію аналогічну до нижнього компоненту навігації в Android, та в яку рекомендовано розміщувати до 5 елементів навігації (рис. 4.6).

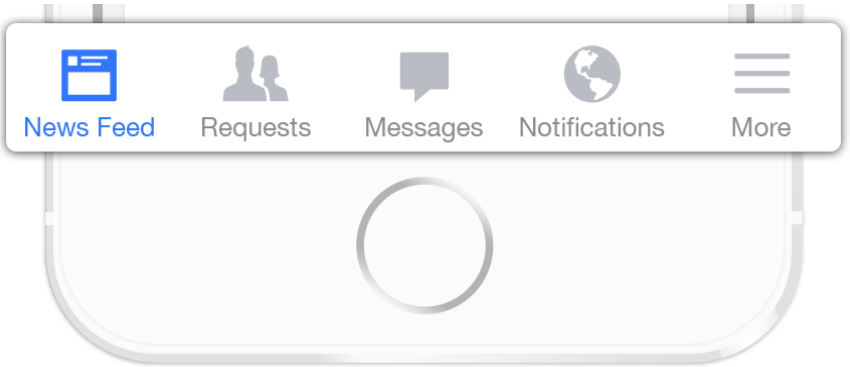

Рисунок 4.6 – Дизайн панелі вкладок на основі AHIG [32]

Дизайн та навігація кнопок в AHIG має відмінний від Material Design підхід, що грунтується на загальній простоті компонентів без використання тіней та анімацій загалом, графічні кнопки IOS пристроїв не мають стандартних для Android пристроїв анімації натискання та зміни стану кнопок. Кнопка AHIG має лише 2 стани, що відповідають станам взаємодії та її відсутності з компонентам (рис 4.7).

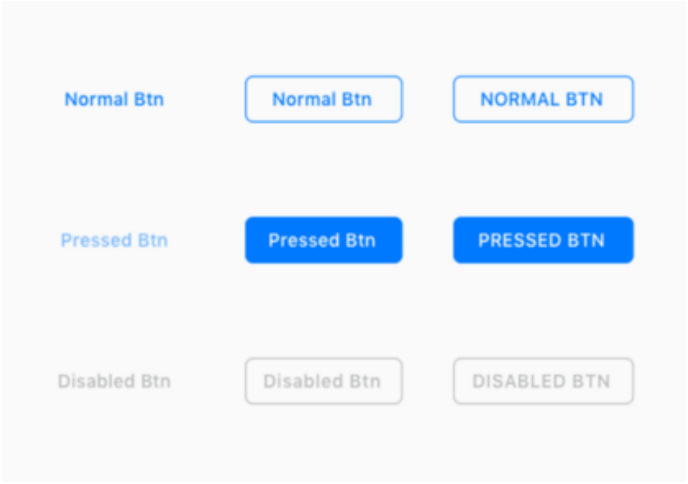

Рисунок 4.7 – Дизайн кнопок на основі AHIG [32]

Операційна система IOS має стандартний шрифт під назвою San Francisco(рис. 4.8).

Рисунок 4.8 – Стандартний шрифт IOS [33]

Загалом AHIG дизайн є чудовим прикладом використання так званого “швейцарського стилю” графічного дизайну, який є протиставленням матеріальному та реалістичному підходу до дизайну графічних інтерфейсів користувача. Компоненти даного підходу позбавляються більшості фізичних властивостей матеріального світу, таких як тінь або об’єм, на заміну котрим

використовуються компоненти з переважанням одного кольору та мінімалістичного дизайну. Завдяки такому підходу до дизайну компонентів відображення та мінімалістичній анімації, знімається додаткове навантаження на роботу процесора та графічного адаптера, що в свою чергу покращує енергоефективність та продуктивність пристрою користувача.

Результатом дослідження є те, що AHIG плоский, дружелюбний та легкий дизайн, що пропагує поступову відмову від скевоморфізму. Material Design в свою чергу розглядає матеріальну складову як метафору та пропонує фізичну, логічну та осмислену анімацію, вважається кросплатформеним. Для дизайну інтерфейсу Android додатку було вирішено взяти принцип Material Design, що органічно виглядає на будь-яких платформах та по стандарту поставляється разом з компонентами відображення Android SDK. Завдяки підтримці мультиплатформеності, додаток органічно виглядатиме на будь-яких операційних системах для яких існують нативні підходи до дизайну графічного інтерфейсу користувача, адже для адаптації додатку необхідно та достатньо переписати лише один модуль додатку.

4.2 Порівняння з конкурентними рішеннями

Результатом виконання став Android додаток, що підтримує подальше мультиплатформене розширення та міграцію базових модулів на інші проєкти. Попереднє внутрішнє тестування отримало позитивні показники по стабільності додатку. Тестувальники відмітили позитивні зміни в порівнянні з конкурентними рішеннями при формуванні запиту. Швидкість формування та подальшого корегування запиту користувача стали набагато зручнішими та інтерактивнішими в порівнянні з рішеннями конкурентних розробок, що зумовленно зміною самого підходу до створення запиту користувачем на користь використання сучасних підходів, а саме делегування розпізнавання та

введення переліку інгредієнтів для запиту за допомогою комп'ютерного бачення (рис. 4.9).

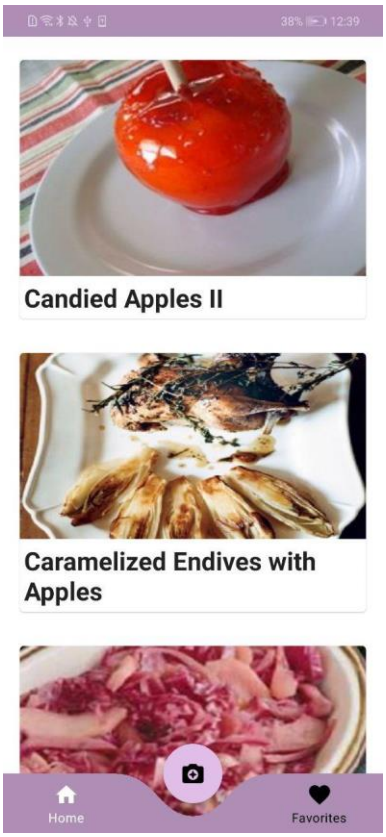

Рисунок 4.9 – Дизайн елементу списку на основі Material Design

Стабільність додатку в порівнянні з конкурентними розробками покращена за допомогою використання сучасних архітектурних підходів та чистоти коду, крім того розробка є нативною для операційної системи на котрій вона використовується, саме тому швидкість відгуку інтерфейсу та загальна стабільність є більш стійкою в порівнянні з рішеннями, що використовують технології кросплатформеності. Для перегляду списку рецептів використовується Android SDK компонент RecyclerView, завдяки якому, такі операції, як прокручування списку рецептів, що призводить до підняття зображень рецептів з кешу, не займають більше 12% процесорного часу на додаток (рис. 4.10).

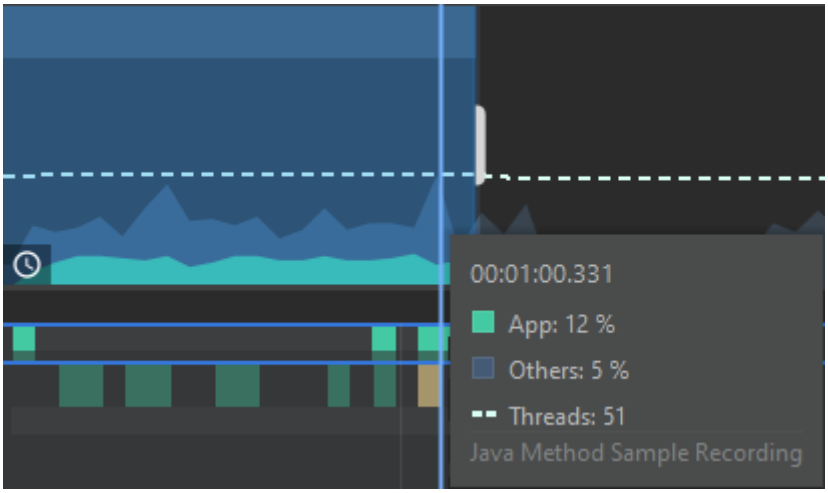

Рисунок 4.10 – Зайнятість CPU під час роботи додатку

Завдяки мінімальному використанню можливостей CPU пристроя користувача, кешуванню результатів запиту, разом зі зображеннями, відсутності сервісів, що працюють паралельно в бекграунді користувача вдалося звести до мінімуму енерговитратність додатку при активній роботі користувача до мінімуму [20] (рис. 4.11).

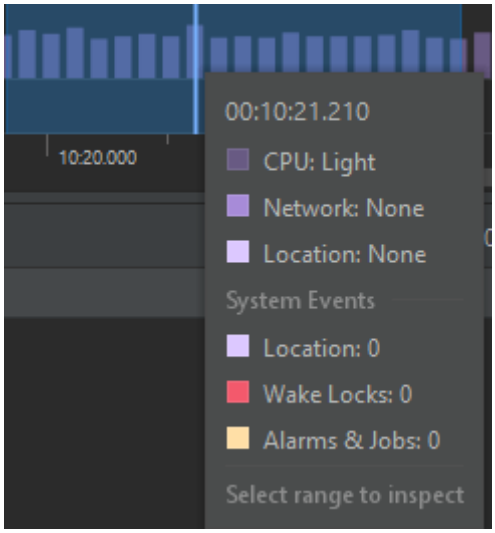

Рисунок 4.11 – Енерговитратність під час роботи додатку

Відмічена можливість використання додатку в офлайн режимі завдяки присутності локальної бази даних користувача, що використовується для зберігання вподобаних рецептів, на відміну від проаналізованих існуючих рішень, що в свою чергу зберігають та дозволяють перегляд рецептів користувача тільки після авторизації на самому сервісі або допоміжних, таких як Facebook, Google Account. В подальшому планується імплементація кешу для запита користувача, що допоможе зекономити інтернет трафік, який витрачається на оброблення запиту по розпізнаванню та отримання результатів з сервісу, що надає рецепти по розпізнаним продуктам.

Дизайн розробки є стандартним для операційної системи Android, відповідно є інтуїтивно зрозумілим для людини, що звикла до використання саме цієї оперативної системи, також для осучаснення дизайну додатку було використано останні версії бібліотек Material Design. Завдяки мультиплатформеності, розширення на інші операційні системи відбуватиметься без зміни шарів бізнес логіки та отримання даних, необхідна зміна виключно шару представлення, що в свою чергу забезпечить можливість використання нативних компонентів дизайну операційної системи на яку відбувається міграція.

Неабиякою перевагою додатку перед конкурентними розробками є швидкість формування запиту користувачем. Наприклад, введення одного лише запиту “apple, orange, banana” з екранної клавіатури, на отримання списку рецептів, займає 11 секунд часу користувача, з використанням сервісу розпізнавання інгредієнтів, час розпізнавання, формування запиту та отримання списку рецептів, займає не більше однієї секунди часу користувача, при достатньому відсотку точності розпізнаваних продуктів (рис. 4.12).

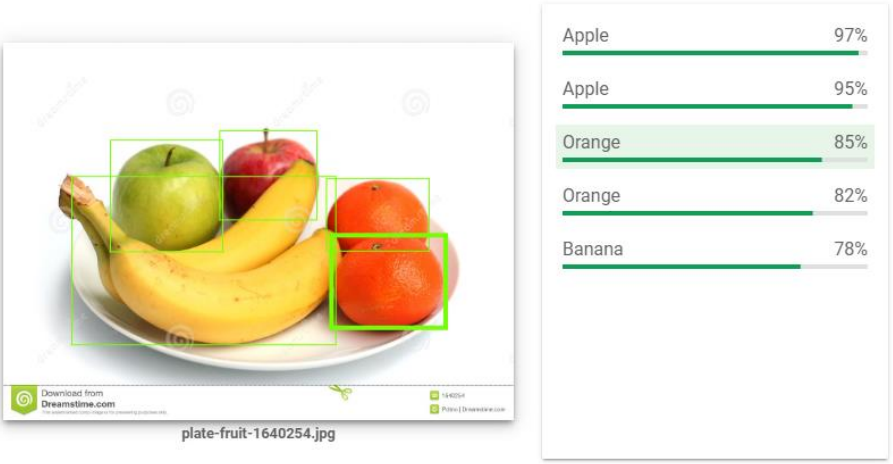

Рисунок 4.12 – Відсоток точності розпізнавання продуктів по фотографії

Однією з основних проблем сучасних Android додатків загалом та представлених рішень зокрема є відсутність можливості збереження стану програми під час зміни орієнтації екрану. Конкурентні розробки вирішують дану проблему шляхом блокування можливості зміни орієнтації взагалі, але це обмежує користувача та не є правильним підходом до розробки програм з різноманітною варіацією екранів з інформацією, що при можливості широкоформатних екранів, мають максимально використовувати вільний простір екрану для розміщення компонентів. Особливо у випадках, коли додаток використовується на планшетах, що мають landscape орієнтацію по стандарту. Завдяки використанню архітектурного шаблону MVP та його базової реалізації бібліотекою Moxy, що додатково реалізує проміжний клас ViewState між імплементаціями Presenter та View, що дозволяє зберігати послідовність викликів API [8], а саме стан додатку перед зміною орієнтації пристрою та фактично зберігається в Presenter, який в свою чергу не знищується під час перевертання на відміну від View, розроблений додаток вирішує цю проблему без обмеження користувача.

4.3 Висновки до розділу

Результатом дослідницької і практичної роботи над проєктом є обрання сервісу для розпізнавання інгредієнтів за зображенням та сервісу, що пропонує рецепти за обмеженою запитом користувача кількістю продуктів. Додаток розроблений на основі аргументованих архітектурних рішень, що дозволяють прозоро відслідковувати кожен етап його роботи, локалізовувати можливі помилки та оперативно їх вирішувати. Підтримується мультиплатформеність продукту, що в подальшому дозволить розширити продукт на інші операційні системи та платформи без зміни фреймворконезалежної частини коду.

Розроблено та впроваджено дизайн, що базується на останніх версіях Material Design, стандартного підходу до дизайну для Android додатків. Програма розроблена на основі початкових вимог, що були сформовані під час аналізу предметної області та скореговані при подальшому аналізі існуючих рішень.

ВИСНОВКИ

Отже, результатом роботи над проєкток є створення додатку, що враховує недоліки існуючих на ринку конкурентів такі, як швидкість формування запиту користувачем, відсутність підтримки роботи зі списком вподобаних користувачем рецептів, загальна стабільність роботи додатку. Кожен недолік існуючих розробок був детально проаналізований та врахований при розробці власного рішення.

Продумана реалізація та оптимізація внутрішніх процесів додатку задовольняє вимоги по енерговитратності та навантаженню на CPU.

Завдяки вдосконаленому методу введення інформації від користувача, що пришвидшує взаємодію та урізноманітнює користувацький досвід був отриманий новий підхід до взаємодії між користувачем та додатком. Як наслідок — форма запиту необхідної інформації від користувача є більш швидкодійною, що в свою чергу економить час, особливо в порівнянні із запропонованими раніше рішеннями на ринку мобільних додатків.

Готовий проєкт, завдяки архітектурним рішенням та вбудованим інструментам мови програмування Kotlin, що дозволяють збирати проєкт на різноманітну кількість платформ, може бути успішно масштабований та мігрований на інші операційні системи без суттєвих змін. Кількість інгредієнтів та якість розпізнавання також піддаються розширенню та покращенню.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Julia. Mobile Application Market Analysis 2020 [Електронний ресурс]. — 2019. — Режим доступу до ресурсу: https://gbksoft.com/blog/mobile-app-market-analysis/.

2 Ann Smarty. 9 Mobile Marketing Trends You Can't Ignore [Електронний ресурс]. — 2019. — Режим доступу до ресурсу: https://www.internetmarketingninjas.com/blog/marketing/mobile-marketing-trends/.

3 Tasty [Електронний ресурс] – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=com.buzzfeed.tasty&hl=ru.

4 GoodFood [Електронний ресурс] – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=uk.co.bbc.goodfood2&hl=ru.

5 SideChef [Електронний ресурс] – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=com.sidechef.sidechef&hl=ru.

6 *Satya Pavan Kantamani*. Detailed Guide on Android Clean Architecture [Електронний ресурс]. — 2019. — Режим доступу до ресурсу: https://medium.com/@pavan.careers5208/detailed-guide-on-android-clean-architecture-9eab262a9011.

7 *Florina Muntenescu*. Android Architecture Patterns Part 2: Model-View-Presenter [Електронний ресурс]. — 2016. — Режим доступу до ресурсу: https://medium.com/upday-devs/android-architecture-patterns-part-2-model-view-presenter-8a6faaae14a5.

8 Moxy [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/Arello-Mobile/Moxy.

9 Cloud Vision API [Електронний ресурс] – Режим доступу до ресурсу: https://cloud.google.com/vision/docs/labels.

10 Spoonacular API [Електронний ресурс] – Режим доступу до ресурсу: https://spoonacular.com/food-api/docs.

11 SQLite [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/SQLite.

12 Room [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/topic/libraries/architecture/room?gclid=Cj0KCQjw2PP1BRCiARIsAEqv-pRxMJcidpLvGrJfTbw_xGqGbW-ox1Zd3aSYL5__MmH1iU_oRsTIFcMaAnouEALw_wcB&gclsrc=aw.ds.

13 Android Architecture [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/topic/libraries/architecture.

14 Mobile databases [Електронний ресурс] – Режим доступу до ресурсу: https://greenrobot.org/news/mobile-databases-sqlite-alternatives-and-nosql-for-android-and-ios/.

15 Material Design Navigation [Електронний ресурс] – Режим доступу до ресурсу: https://material.io/design/navigation/understanding-navigation.html#lateral-navigation.

16 Martin R. Clean Architecture: A Craftsman's Guide to Software Structure and Design / Robert Cecil Martin, 2018. – P. 201–211.

17 Reenskaug T. The DCI Architecture: A New Vision of Object-Oriented Programming b [Електронний ресурс] / T. Reenskaug, J. O. Coplien. – 2009. – Режим доступу до ресурсу: https://www.artima.com/articles/dci_vision.html.

18 Alliaume E. Hexagonal Architecture: three principles and an implementation example [Електронний ресурс] / E. Alliaume, S. Roccaserra. – 2018. – Режим доступу до ресурсу: https://blog.octo.com/en/hexagonal-architecture-three-principles-and-an-implementation-example/.

19 Entity-control-boundary architecture [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Entity-control-boundary.

20 An Energy Efficient Android Application [Електронний ресурс] / V.Hurbungs, Y. Beeharry, G. Ahotar, A. Calkee. – 2016. – Режим доступу до ресурсу:

https://pdfs.semanticscholar.org/33d2/1599e378486e7d20fcee0e5b585589be356b.pdf
.

21 Edamam API [Електронний ресурс] – Режим доступу до ресурсу: https://developer.edamam.com/edamam-recipe-api.

22 BigOven API [Електронний ресурс] – Режим доступу до ресурсу: https://api2.bigoven.com/web/documentation/recipes.

23 Microsoft Azure Cognitive Services [Електронний ресурс] – Режим доступу до ресурсу: https://azure.microsoft.com/ru-ru/services/cognitive-services/computer-vision/#features.

24 Amazon Rekognition API [Електронний ресурс] – Режим доступу до ресурсу: https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/what-is.html.

25 Watson Visual Recognition [Електронний ресурс] – Режим доступу до ресурсу: https://cloud.ibm.com/docs/visual-recognition?topic=visual-recognition-getting-started-tutorial#tutorial-de-introdu-o.

26 Lecun Y. Convolutional Networks for Images, Speech, and Time-Series [Електронний ресурс] / Y. Lecun, Y. Bengio. – 1995. – Режим доступу до ресурсу: http://yann.lecun.com/exdb/publis/pdf/lecun-bengio-95a.pdf.

27 Martin R. C. The Clean Code Blog [Електронний ресурс] / Robert C. Martin. – 2012. – Режим доступу до ресурсу: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html.

28 Clean Architecture Diagram [Електронний ресурс] – Режим доступу до ресурсу: https://android.jlelse.eu/thoughts-on-clean-architecture-b8449d9d02df.

29 Clean Architecture Android Diagram [Електронний ресурс] – Режим доступу до ресурсу: https://www.fandroid.info/lektsiya-4-po-arhitekture-android-prilozheniya-clean-arcitecture/.

30 Moxy Diagram [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/redmadrobot-mobile/android-without-lifecycle-mpvsv-approach-with-moxy-6a3ae33521e.

31 Карпович А. В. Використання згорткових нейронних мереж для задачі класифікації текстів [Електронний ресурс] / Артем Валерійович Карпович // International scientific journal «Internauka». – 2018. – Режим доступу до ресурсу: http://nbuv.gov.ua/j-pdf/mnj_2018_14_18.pdf.

32 Геллар Н. Различие в проектировании нативных приложений iOS и Android [Електронний ресурс] / Николай Геллар. – 2018. – Режим доступу до ресурсу: https://ux.pub/razlichie-v-proektirovanii-nativnyx-prilozhenij-ios-i-android/

33 Абраров А. 32 отличия дизайна мобильного приложения под iOS и Android [Електронний ресурс] / Артур Абраров. – 2019. – Режим доступу до ресурсу: https://vc.ru/design/93884-32-otlichiya-dizayna-mobilnogo-prilozheniya-pod-ios-i-android#:~:text=%D0%94%D0%B8%D0%B7%D0%B0%D0%B9%D0%BD%20iOS%2D%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F%20%D1%81%D0%BE%D0%B7%D0%B4%D0%B0%D1%91%D1%82%D1%81%D1%8F%20%D0%B2,%2C%20xhpdi%2C%20xxhdpi%20%D0%B8%20xxxhdpi.

34 Применение графовых баз данных [Електронний ресурс] – Режим доступу до ресурсу: https://nitrosdata.ru/2019/02/20/primenenie-grafovyh-baz-dannyh/.

35 Gallinucci E. Schema profiling of document-oriented databases / E. Gallinucci, M. Golfarelli, S. Rizzi. // Informantion Systems. – 2018. – №75. – P. 13–25.